-EXPRESS MAIL NO. EV265280662US

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 714 213 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
06.03.2002 Bulletin 2002/10

(51) Int Cl.7: **H04N 7/50**, H04N 5/00, H04N 7/52

(21) Application number: **95308305.2**

(22) Date of filing: **21.11.1995**

(54) **MPEG2 transport decoder**

Transportdekodierer füu MPEG2

Décodeur de transport pour MPEG2

(84) Designated Contracting States:
DE FR GB NL

(30) Priority: **23.11.1994 KR 3087594**

(43) Date of publication of application:
29.05.1996 Bulletin 1996/22

(73) Proprietor: **LG ELECTRONICS INC.**
Seoul (KR)

(72) Inventors:
- **Rim, Chai Yeol**
Seodaemun-ku, Seoul (KR)
- **Lee, Hyun Soo**
Sungdong-ku, Seoul (KR)

(74) Representative:
Cross, Rupert Edward Blount et al
BOULT WADE TENNANT, Verulam Gardens 70
Gray's Inn Road
London WC1X 8BT (GB)

(56) References cited:
EP-A- 0 679 028 **US-A- 5 148 272**

- **STAMMNITZ P ET AL: "HARDWARE IMPLEMENTATION OF THE TRANSPORT STREAM DEMULTIPLEXER FOR THE HDTVT DEMONSTRATOR" SIGNAL PROCESSING OF HDTV. PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON HDTV, 26 - 28 October 1994, pages 435-441, XP002046490**

EP 0 714 213 B1

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates to an MPEG2 transport decoder, more particularly to an MPEG2 transport decoder which is programmable for multipurpose use.

[0002] Recently, as for a format for transmitting/receiving a digitally processed picture and an audio between media, a number of methods are suggested. Among them, there is an MPEG2 system part suggested from the moving picture experts group MPEG2 so that the data can be transmitted and received between media with being made as a format convenient for using the compressed picture and sound data. The transmitting/receiving format is divided into two types: one is to transmit/receive in an error free environment as of a medium of storing device; the other is to transmit/receive in an environment in which an error may be occurred as of a medium of a satellite or a cable. The transmitting/receiving in an environment in which an error never occurs is performed by formatting such manner of a program stream and in an environment in which an error may often occur is performed by formatting such manner of a transport packet stream.

[0003] A typical MPECG2 transport decoder applied in a transmitting/receiving device in an error having environment is divided into the cases of using a central processor unit CPU and using a hardwired logic.

[0004] In the MPEG2 transport decoder using the CPU, the operation of the CPU should be performed at a high speed and in the MPEG2 transport decoder using the hardwired logic, the use should be fixed in a certain purpose.

[0005] Referring to the attached drawings, conventional construction of the MPEG2 transport decoder is described below.

[0006] Fig. 1 illustrates the MPEG2 transport decoder using a conventional CPU. Fig. 2 illustrates the MPEG2 transport decoder using a conventional hardwired logic. The MPEG2 transport decoder using a conventional CPU as illustrated in Fig.1, comprises a channel decoding unit 1 outputting a transport packet data by tuning and demodulating a signal received through a satellite or a cable, a data buffer unit 2 outputting after storing momentarily a transport packet data outputted from the channel decoding unit 1; a CPU 3 performing a decoding operation as programmed in the memory by reading a data outputted from the data buffer unit 2; a memory unit 4 storing a program to be operated by the CPU 3; and three decoders of a video decoder 5, an audio decoder 6, and a data decoder 7 decoding a video signal, an audio signal and a data signal each with the CPU 3. On the other hand, the MPEG2 transport decoder using a conventional CPU as illustrated in Fig. 2, comprises a channel decoding unit 1 outputting a transport packet data by tuning and demodulating a signal received through a satellite or a cable, a hardwired logic unit 8 decoding a transport packet data outputted from the channel decoding unit 1 in such a manner of hardwiring; and three decoders of a video decoder 5, an audio decoder 6, and a data decoder 7 decoding a video signal, an audio signal and a data signal each with the hardwired logic unit.

[0007] As described above, the conventional transport decoding is performed only with the CPU or the hardwired logic. That is, as of Fig. 1, as CPU is used for encompassing all various applications, the CPU reads a transport packet data from the channel decoder' unit 1, performs a decoding 'operation according to the programmed on the memory unit 4 and outputs the decoded data to a video, an audio, and a data decoder. However, with these systems in which the data is processed by programming, a high-speed CPU is needed to perform a high-speed decoding operation. And as illustrated in Fig. 2, in case of composing with a hardwired logic, a decoding operation is possible in the fixed applications, however in case of different applications, the decoding operation is not flexible. In addition, when using an unfixed field or a private data is inputted thereto, if the related condition is not composed of hardwired logic, a hardwired logic circuit should be recomposed to process the case.

[0008] EP0679028A2 which forms part of the "state-of-the-art" according to Art 54(3) EPC, describes an inverse transport processor system for a TDM packet signal TV receiver which includes apparatus for selectively extracting desired payloads of program component data and coupling this data to a common buffer memory data input port. A microprocessor (19) associated with the system also couples data to the common buffer memory (18) data input port. The respective component payloads and data generated by the microprocessor are stored in respective blocks of the common buffer memory in response to associated memory address which are applied to a memory address input port by an address multiplexer (17). A decryption device (16) is included to decrypt payload data according to packet specific decryption keys. In addition a detector (15) is included to detect payloads including entitlement data. Payloads containing entitlement data are directed,via the common buffer memory to a smart card which generates the packet specific decryption keys. A memory data output port is coupled to a bus interconnected with the respective program component processors (21-24). Responsive to data requests from the respective program component processors, and data write requests from the component payload source, memory access for read and write functions is arbitrated (17) so that no incoming program data is lost.

SUMMARY OF THE INVENTION

[0009] Particular embodiments of the present invention seek to address the problems of prior art, so that

the transport decoding operation may be variously applied according to the programmed contents by user by composing the MPEG2 transport decoder with incorporating a hardwired logic and a CPU capable of being programmed. To this end in a currently preferred embodiment the MPEG transport decoder comprising: a channel decoder unit for outputting a signal received through a satellite or a cable tuned to receive transport packet data; a transport decoder for decoding the transport packet data; and video (5), audio (6), and data decoders (7) for decoding video, audio and data signals via the transport decoder, said transport decoder further comprising:

a transport parser unit (11) for storing at least one syntax field value in at least one packet decoder register, said syntax field value obtained by parsing said transport packet data, for outputting said transport packet data, whereby each packet of said transport packet data is identified using a packet identifier (PID) obtained from said received transport packet data, and for outputting an interrupt signal if a pre-defined syntax field value is stored within any one of said packet decoder registers;
a CPU interface unit (14) for providing an interface between said at least one packet decoder register of said transport parser unit (11) and each of said video (5), audio (6) or data (7) decoders, for outputting a signal selecting either the transport parser unit (11) or the video (5), audio (6) or data (7) decoders or a first memory unit (22) the selection being made by decoding an address, the CPU interface unit further comprising at least one interrupt register for receiving at least one interrupt signal from either of said transport parser unit (11) or said video (5), audio (6) or data (7) decoders;
a CPU (13) for reading the contents of said interrupt register after the interrupt signal has been inputted from either the transport parser unit (11), the video (5), audio (6) or data decoder (7), for detecting whether the interrupt signal is inputted from said transport parser unit (11) or from the video (5), audio (6) or data decoder (7), and arbitrating CPU (13) access to said transport parsing unit (11), video (5), audio (6) or data decoders (7) according to a program which is stored on a second memory unit (12), and which is arranged to determine the operations of said CPU (13); and
a decoder interface unit (15) for controlling the order of exchange of said transport packet data amongst said CPU (13), said transport parser unit (11), or said video (5), audio (6) or data decoders (7).

BRIEF DESCRIPTION OF THE DRAWINGS

[0010]

Fig. 1 is a configuration view of an MPEG2 transport decoder using a conventional CPU;
Fig. 2 is a configuration view of an MPEG2 transport decoder using a conventional hardwired logic;
Fig. 3 is a configuration view of an MPEG2 transport decoder capable of being programmed according to a preferred embodiment of the invention;
Fig. 4 is a detailed block diagram of a transport parser unit and a CPU interface unit of Fig. 3;
Fig. 5 is a detailed block diagram of an embodiment of a decoder interface unit of Fig. 4;
Fig. 6 is a detailed block diagram of another embodiment of a decoder interface unit of Fig. 4;
Fig. 7 illustrates a transport packet syntax inputted to a transport packet decoder resister according to a preferred embodiment of the invention;
Fig. 8 illustrates a ADF syntax inputted to a ADF decoder resister according to a preferred embodiment of the invention;
Fig. 9 illustrates a PES packet syntax inputted to a PES decoder register according to a preferred embodiment of the invention;
Figs. 10A, 10B and 10C illustrate a PSI syntax inputted to a PSI decoder register according to a preferred embodiment of the invention;
Fig. 11 illustrates a register list of the transport decoder according to a preferred embodiment of the invention;
Fig. 12 illustrates a register list of a ADF decoder according to a preferred embodiment of the invention;
Fig. 13 illustrates a register list of a PES decoder according to a preferred embodiment of the invention;
Fig. 14 illustrates a table of interrupt generation in a transport decoder according to a preferred embodiment of the invention;
Fig. 15 illustrates a table of interrupt generation in a ADF decoder according to a preferred embodiment of the invention;
Fig. 16 illustrates a table of interrupt generation in a PES decoder according to a preferred embodiment of the invention; and
Fig. 17 illustrates a table of interrupt generation in a CPU controlling interface unit according to a preferred embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0011] An MPEG2 transport decoder embodying the present invention is now described with reference to the attached drawings.

[0012] As illustrated in Fig. 3, An MPEG2 transport decoder according to the regulations of the MPEG2 system comprises a transport parser unit 11 storing each syntax field value after parsing, outputting each audio, video and data information which is set of PID after collecting from each packet data inputted from the channel

and outputting the interrupt signal if an identified register value of the register values is set, a CPU interface unit 14 providing an interface between the register file of the transport parser unit 11 and each decoder, and outputting a signal selecting a transport parser unit 11 or a video decoder, an audio decoder, a data decoder and a memory by decoding the address, a CPU 13 reading the interrupt register from the CPU interface unit once an interrupt signal is inputted, detecting if the interrupt signal is inputted from the transport parser unit 11 or from a video decoder, the audio decoder and the data decoder, and decoding according to the program on a first memory unit 12, the first memory unit 12 storing a program of the operations of CPU 13, and a decoder interface unit 15 controlling in order to exchanging the data among the CPU 13, the transport parser unit 11, and the video, audio, data decoders.

[0013] The figures which is not described in Fig. 3 is provided as follows.

[0014] A data1 is a data bus of CPU 13, an addr1 is an address bus of CPU, and a cntrll is a controlling signal of CPU and is composed of a READ/WRITE signal, a STROBE signal, a READY signal and an interrupt signal.

[0015] A data2 is a data bus, a cntrl2 is composed of the selecting signal to identify the registers of the transport parser unit 11, read the value with the data bus 2 or to write the value of the data 2 bus, a selecting signal to read and write the audio, video and another decoders, a strobe signal and a ready signal.

[0016] A data3 is a data bus inputted from the channel decoder and a cntrl3 is composed of a Read Enable signal to read the data from the channel decoder, Read Clock, and a Ready signal.

[0017] A data4 is composed of the video, audio and data buses to input/output to the video, audio and data decoders, an addr4 is composed of an address bus of the video, audio and data addresses to output to the video, audio and data decoders, and a cntrl4 is composed of a controlling signal to read/write of the video, audio and data decoders.

[0018] A data5 is a data bus transmitting the data between the transport parser unit 11 and the decoder interface unit 15, a cntrl5 is a controlling signal bus transmitting the controlling signal between the PES decoder 24 of the transport parser unit 11 and the decoder interface unit 15.

[0019] Thus-composed MPEG2 transport decoder embodying the invention is now described in detail as follows.

[0020] In Fig. 4, the transport parser unit 11 is comprised of a channel decoder interface unit 20, a transport decoder 21, a second memory unit 22, an adaptation field ADF decoder 23 and a packetized elementary stream PES decoder 24. That is, the channel decoder interface unit 20 is an interface for inputting/outputting the data and the controlling signal with the channel decoder unit 1 in the transport parser unit 11. The transport decoder 21 is comprised of a transport packet decoder controller 21a for parsing the MPEG2 transport packet syntax and a transport packet decoder register 21b for storing each header field value parsed in the transport packet decoder controller 21a. The registers in the transport decoder register 21b are accessed by the CPU 13 and the parsed field value is interrupted according to the enabling state of the interrupt register to the CPU 13.

[0021] The second memory unit 22 is comprised of a memory 22a for storing the PSI sections of the MPEG2 streams, and the adaptation_extension_data and the transport _private_data shown in Fig. 8 and the PES_extension_data and the DSM_trick_mode_data shown in Fig. 9 contained in the selected packet, and a memory controller 22b for producing storage addresses mem_addr at which those data to be stored using information on memory address indicating storage positions of those data and producing cntrl_mm to control memory access. The memory 22a may include a DRAM or an SRAM, and the memory controller 22b has start addresses and end addresses required for storing each PSI section and four kinds of data in the memory 22a and a write address for writing. The CPU 13 may designate the start addresses and the end addresses, and the write address serves to store the data started from the start addresses to the end addresses increasing the addresses automatically.

[0022] The ADP decoder 23 is comprised of an ADF decoder controller 23a for parsing the ADF data of the MPEG2 transport packet syntaxes and an ADF decoder register 23b for storing each field value of the parsed header in the ADF decoder controller 23a. The registers in the ADF decoder registers 23b are accessed by CPU 13 and the parsed field value is interrupted according to the enabling state of the interrupt register to the CPU 13.

[0023] The PES decoder 24 is comprised of a PES decoder controller 24a for parsing the PES section of the MPEG2 streams and a PES decoder register 24b for storing each field value of the parsed header. The registers in the PES decoder registers 24b are accessed by CPU 13 and the parsed field value is interrupted according to the enabling state of the interrupt register to the CPU 13.

[0024] On the other hand, the CPU interface unit 14 is comprised of a data buffer 36, a CPU address decoder 31, a tp-CPU interface unit 32, a memory interface unit 33, an adf-CPU interface unit 34 and a pes-CPU interface unit 35. Namely, the data buffer 36 performs a buffering to read/write the contents of the CPU data bus. The CPU address decoder 31 generates a selecting signal for selecting a register of the transport parser unit 21 by decoding the high address part of the CPU 13, a selecting signal for accessing the video, audio and another decoders and a selecting signal for accessing the program/data memory. The tp-CPU 32 generates a controlling signal cntrl-dsp-td for the CPU 13 to access the register in the transport packet decoder register 21b by incorporating a controlling signal cntrll of the CPU 13,

an address signal addr1 and a selecting signal, and decodes each address for each register to have other address. The memory interface unit 33 generates a controlling signal cntrl-dsp-mem for the CPU 13 to access the register in the memory controller 22b by incorporating a controlling signal cntrll of the CPU 13, an address signal addr1 and a selecting signal, and decodes each address for each register to have other address. The adf-CPU interface unit 34 generates a controlling signal cntrl-dsp-adf for the CPU 13 to access the register in the ADF decoder register 23b by incorporating a controlling signal cntrl1 of the CPU 13, an address signal addr1 and a selecting signal, and decodes each address for each register to have other address. The pes-CPU interface unit 35 generates a controlling signal cntrl-dsp-pes for the CPU 13 to access the register in the PES decoder register 24b by incorporating a controlling signal cntrll of the CPU 13, an address signal addr1 and a selecting signal, and decodes each address for each register to have other address.

[0025] The decoder interface unit 15 is comprised of a video decoder interface unit 41, an audio decoder interface unit 42 and a data decoder interface unit 43.

[0026] The video decoder interface unit 41 controls an address bus, a data bus and a controlling signal for the CPU 13 and the PES decoder 24 to access the video decoder jointly. In other words, the video decoder interface unit 41 receives an address bus signal addr1, a data bus signal data2 and a video decoder selecting signal from the CPU 13 and inputs or outputs the video address, the video data and controlling signals, thereby accessing the video decoder by the CPU 13. And after storing momentarily the data signal data5 and the controlling signal controls from the PES decoder 24, the video decoder interface unit 41 outputs the video data, the video control and the video address to the video decoder 5 while the CPU 13 does not access the video decoder 5.

[0027] The audio decoder interface unit 42 controls an address bus, a data bus and a controlling signal for the CPU 13 and the PES decoder 24 to access the audio decoder jointly. In other words, the audio decoder interface unit 42 receives an address bus signal addr1, a data bus signal data2 and a audio decoder selecting signal from the CPU 13 and inputs or outputs the audio address, the audio data and controlling signals, thereby accessing the audio decoder 6 by the CPU 13. And after storing momentarily the data signal data5 and the controlling signal controls from the PES decoder 24, the audio decoder interface unit 42 outputs the audio data, the audio control and the audio address to the audio decoder 6 while the CPU 13 does not access the audio decoder 6.

[0028] The data decoder interface unit 43 controls an address bus, a data bus and a controlling signal for the CPU 13 and the PES decoder 24 to access the data decoder jointly. In other words, the data decoder interface unit 43 receives an address bus signal addr1, a data bus signal data2 and a data decoder selecting signal from the CPU 13 and inputs or outputs the data address, the data and controlling signals, thereby accessing the data decoder 7 by the CPU 13. And after storing momentarily the data signal data5 and the controlling signal controls from the PES decoder 24, the data decoder interface unit 43 outputs the data, the data control and the data address to the data decoder 7 while the CPU 13 does not access the data decoder 7.

[0029] On the other hand, the decoder interface unit 15 illustrated in Fig. 4 is comprised of the video decoder interface unit 41, the audio decoder interface unit 42 and the data decoder interface unit 43 and each decoder interface unit 41, 42 and 43 has the same configuration as of Fig. 5, however need not be composed all at once and may be partially composed according to the application.

[0030] Therefore in Fig. 5 only the video decoder interface unit 41 is illustrated. That is, the video decoder interface unit 41 comprises a first data buffer 51 outputting after storing momentarily the data outputted from the CPU 13, a FIFO 52 storing momentarily the data outputted from the transport parser unit 11 and then outputting firstly the data inputted before, simultaneously with outputting the signals fifo-ef and fifo-ff indicating if the data is filled out or not, a second data buffer 53 outputting the data outputted from the FIFO 52 after storing momentarily, an access controlling unit 54 setting the signal token giving right of an access to the CPU 13 according to the accessing condition of the video decoder 5 by the outputting signals fifo-ef and fifo-ff of the FIFO 52 and the CPU 13, and an interface controlling unit 55 finishing the present accessing work with the signal token from the access controlling unit 54 and controlling for the CPU 13 to access to read/write the video decoder.

[0031] Fig. 6 is a detailed block diagram of another embodiment of the decoder interface unit of Fig. 4.

[0032] In an embodiment of Fig. 5, a FIFO is installed in each video, audio and data decoder interface unit 41, 42 and 43 and the FIFO is installed in common in other embodiment of Fig. 6.

[0033] That is, it is comprised of a first data buffer 51 outputting after storing momentarily the data outputted from the CPU 13 to the video decoder interface unit 41, the audio decoder interface unit 42 and the data decoder interface unit 43, a second data buffer 53 outputting the decoder data after storing momentarily, an access controlling unit 54 setting the signal token giving right of an access to the CPU 13 according to the accessing condition of the video decoder 5 with the signals fifo-ef and fifo-ff outputted through a controlling signal line vid-mem-cntrl and the CPU 13, and an interface controlling unit 55 finishing the present accessing work with the signal token from the access controlling unit 54 and controlling for the CPU 13 to access to read/write the video decoder. And the common memory unit 44 is installed in the video decoder interface unit 41, the audio decoder

interface unit 42 and the data decoder interface unit 43. Namely, the memory unit 44 comprises a third data buffer 61 storing the output data of the transport parser unit 11 momentarily, a memory 68 divided into three areas of video, audio and data storing areas and storing the data inputted from the third data buffer 61 or outputting the stored data into the decoder data, a video writing pointer 62 outputting a writing address in order to write the video data to the memory 68, an audio writing pointer 63 outputting a writing address in order to write the audio data to the memory 68, a data writing pointer 64 outputting a writing address in order to write the data to the memory 68, a first address buffer 65 storing momentarily the address outputted from the video writing pointer 62, a second address buffer 66 storing momentarily the address outputted from the audio writing pointer 63, a third address buffer 67 storing momentarily the address outputted from the data writing pointer 64, a video reading pointer 72 outputting a reading address in order to read the video data to the memory 68, an audio reading pointer 73 outputting a reading address in order to read the audio data to the memory 68, a data reading pointer 74 outputting a reading address in order to read the data to the memory 68, a fourth address buffer 69 storing momentarily the address outputted from the video reading pointer 72, a fifth address buffer 70 storing momentarily the address outputted from the audio reading pointer 73, a sixth address buffer 71 storing momentarily the address outputted from the data reading pointer 74, and a memory interface controller 75 controlling the operations of the pointers 62, 63, 64, 72, 73 and 74 and the address buffers 65, 66, 67, 69, 70 and 71 by the controlling signal cntrl5 from the PES decoder 24.

[0034] An operation of the embodiment composed as described above is as follows.

[0035] Fig. 7 illustrates a transport packet syntax inputted to a transport packet decoder register 21b; Figs. 8A and 8B illustrate the ADF syntax inputted to a ADF decoder register 23b; Figs. 9A, 9B and 9C illustrate the PES packet syntax inputted to a PES decoder register 24b; and Figs. 10A, 10B and 10C illustrate the PSI syntax to be stored in the memory 22a.

[0036] And Fig. 11 illustrates a register list of the transport decoder 21; Fig. 12 illustrates a register list of a ADF decoder 23; and Fig. 13 illustrates a register list of a PES decoder 24.

[0037] Additionally, Fig. 14 illustrates a table of interrupt generation in a transport decoder 21; Fig. 15 illustrates a table of interrupt generation in a ADF decoder 23; Fig. 16 illustrates a table of interrupt generation in a PES decoder 24 according to a preferred embodiment of the invention; and Fig. 17 illustrates a table of interrupt generation in a CPU controlling interface unit 14.

[0038] First, the transport parser unit 11 storing each syntax field value to the register after parsing according to the standards of the MPEG2 system, outputting each audio, video and data information which is set of PID after collecting from each packet data inputted from the channel and outputting the interrupt signal if an identified register value of the register values is set of each transport, ADF, PES and PSI decoder. That is the transport decoder 21 of the transport parser unit 11 parses the MPEG2 transport packet syntax as illustrated in Fig. 7 in the transport packet decoder controller 21A and stores the MPEG2 transport packet syntax to the transport packet decoder register 21B as illustrated in Fig. 11. And the field value of the transport packet decoder register 21B generates an interrupt to the CPU 13 according to the state of the interrupt register enable.

[0039] An interrupt generating method is that as illustrated in Fig. 14, after the PID field in the transport packet head is compared with the PID in the packet which the user wants, once the compared values are equal with each other, the outcome of the comparison is 1; once the compared values are different with each other, the outcome is 0. In such a manner that it is compared as follows: for the PID-V-flag, the transport PID field is compared with the PID of the video packet and the output that the compared values are same is equal to 1 while that of the different values is equal to 0; for the PID-A-flag, the transport PID field is compared with the PID of the audio packet and the output that the compared values are same is equal to 1 while that of the different values is equal to 0; for the PID-D-flag, the transport PID field is compared with the PID of the data packet and the output that the compared values are same is equal to 1 while that of the different values is equal to 0; for the PID-PAT-flag, the transport PID field is compared with the PID of the program association table PAT and the output that the compared values are same is equal to 1 while that of the different values is equal to 0; for the PID-PMT-flag, the transport PID field is compared with the PID of the program map table PMT and the output that the compared values are same is equal to 1 while that of the different values is equal to 0; for the PID-CAT-flag, the transport PID field is compared with the PID of the conditional access table CAT and the output that the compared values are same is equal to 1 while that of the different values is equal to 0; and for the PID-NIT-flag, the transport PID field is compared with the PID of the network information table NIT and the output that the compared values are same is equal to 1 while that of the different values is equal to 0.

[0040] The ADF decoder 23 parses the MPEG2 ADF field syntax as illustrated in Fig. 8 in the ADF decoder controller 23A and stores the MPEG2 ADF field syntax in the ADF decoder register 23B as illustrated in Fig. 12. And the field value of the ADF decoder register 23B generates an interrupt to the CPU 13 according to the state of the interrupt register enable. In this time, the manner of interrupt generation is as shown in Fig. 15.

[0041] The PES decoder 24 parses the MPEG2 PES field syntax as illustrated in Fig. 9 in the PES decoder controller 24A and stores the MPEG2 PES field syntax in the PES decoder register 24B as illustrated in Fig. 12. And the field value of the PES decoder register 24B gen-

erates an interrupt to the CPU 13 according to the state of the interrupt register enable. The interrupt generating method is illustrated in Fig. 16.

[0042] The decoder interface unit 15 of Fig. 3 is as follows:

[0043] First, the video decoder interface unit 41 receives an interrupt from the video decoder 5, generates a video interrupt signal Vid-int and transmits the Vid-int through the controlling bus cntrl12 to the CPU controlling interface. The audio decoder interface unit 42 receives an interrupt from the audio decoder 6, generates an audio interrupt signal Aud-int and transmits the Aud-int through the controlling bus cntrl12 to the CPU controlling interface. The data decoder interface unit 43 receives an interrupt from the data decoder 7, generates a data interrupt signal Data-int and transmits the Data-int through the controlling bus cntrl12 to the CPU controlling interface.

[0044] An example in which the configuration of the decoder interface unit 15 is as illustrated in Fig. 5 is now described in detail.

[0045] The video decoder interface unit 41 receives an address bus signal addr2, a data bus signal data2 and a video decoder selecting signal from the CPU 13 through the selecting signal line, input/outputs the video address, video data and the video controlling signal and has the CPU 13 access the video decoder 5. And the video decoder interface unit 41 receives and stores momentarily a data signal data5 and a controlling signal cntrl5 from the PES decoder 24 and has the video decoder 5 output the video data, the video control and the video address while the CPU 13 does not access the video decoder so that the data bus signal data2 from the CPU 13 is momentarily stored in the first data buffer 51 and the data signal data5 and the controlling signal cntr15 from the PES decoder 24 is also momentarily stored in the FIFO 52, thereby being outputted through the second data buffer 53. In this time, the FIFO 52 detects if the data is full or empty and outputs the related signals fifo-ef and fifo-ff. The access controlling unit 54 detects if the CPU 13 accesses the video decoder 5 with the received signals fifo-ef and fifo-ff from the FIFO 52 and the controlling signal cntrl2 of CPU 13 and sets the signal token giving right of access to the CPU 13 if the CPU 13 accesses the video decoder. Accordingly, when the signal is changed from 0 to 1, the mode is also changed for the CPU 13 to access the video decoder 5 from the state of transmitting the fifo data to the video decoder 5 so that the interface controller 55 finishes the present accessing job completely and controls the CPU 13 to access the video decoder 5.

[0046] When the CPU 13 finishes accessing and after a predetermined period of delaying time, the access of the CPU 13 does not exist and the output signal token from the access controlling unit 54 is reset for 0, the interface controller 55 is operated in the FIFO data transfer mode.

[0047] The operation is performed in the same manner in the video decoder interface unit 41, the audio decoder interface unit 42 and the data decoder interface unit 43.

[0048] While, the configuration of the decoder interface unit 15 is as illustrated in Fig. 6 in which three FIFO of Fig. 5 are employed as a memory is operated as follows.

[0049] First, the video, audio and data decoder interface units 41, 42 and 43 are operated as illustrated in Fig. 5. The memory unit 44 uses a video reading pointer 72 and a video writing pointer 62 in order to access the video data for performing the operation of FIFO, uses an audio reading pointer 73 and an audio writing pointer 63 to access the audio data and uses a data reading pointer 74 and a data writing pointer 64. The video data accessing pointers 72 and 62 increase the reading/writing pointer by 1 point after reading/writing for one time to access the video memory area and are increased by 1 point with returning to the first address after accessing the boundary section of the video memory area. And the audio and data accessing pointers 73, 63, 74 and 64 are also operated in the same manner of the video data accessing pointers.

[0050] The value of each pointer 62, 63, 64, 72, 73 and 74 is applied to the memory through each address buffer 65, 66, 67, 69. 70 and 71 and each address buffer 65, 66, 67, 69, 70 and 71 is output-enabled by the memory interface controller 75 and applied to the address port of the memory 68. Therefore, the data data5 outputted from the transport parser unit 11 is applied through the third data buffer 61 to the memory 68 in case of the writing enable and in case of reading, the identified value by each reading pointer 72, 73 and 74 is outputted as a decoder data. The memory interface controller 75 receives and transmits the controlling signal cntrl5 from the transport parser unit 11 and the reading controlling signal vid-mem-cntrl, aud-mem-cntrl and data-mem-cntrl from the video, audio and data decoder interface units 41, 42 and 43, applies the signals to the memory 68 according to the sorts of data identified by the controlling signal cntrl5 among the video, audio and data writing data in order to read the data of the transport parser unit 11 to the memory 68, controls to perform the operation of writing the data data5 to the memory 68, receives the controlling signals vid-mem-cntrl, aud-mem-cntrl and data-mem-cntrl from each decoder interface unit 41, 42 and 43, applies each reading pointer 72, 73 and 74 to the memory 68 while the data is not inputted from the transport parser unit 11, reads the data and outputs the data to each decoder interface units 41, 42 and 43. The operation of CPU interface unit 14 of Fig. 3 provides the register file, an interface between the video, audio decoders and another decoders of the transport parser unit 11 and outputs a selecting signal selecting one among the transport parser unit 11 or the video, audio and data decoders 5, 6 and 7 and the first memory unit 12 by decoding the address. That is, the CPU address decoder 31 decodes a high-address part of the

CPU 13 and outputs the selecting signal to select a register of the transport parser unit 11 and a selecting signal to access the program/data memory. And the tp-CPU interface unit 32 decodes the address for each register to have other address by incorporating the controlling signal cntrl1, the address signal addr1 and the selecting signal because that the CPU 13 generates a signal cntrl-dsp-td to access the register on the transport packet decoder register 21B. In another mem-CPU interface unit 33, adf-CPU interface unit 34 and pes-CPU interface unit 35, as the CPU 13 generates the signals cntrl-dsp-mem, cntrl-dsp-adf and cntrl-dsp-pes to access the corresponding register of the transport parser unit 11 as in the tp-CPU interface unit 32, the controlling signal cntrll, the address signal addr1 and the selecting signal are incorporated to decode the address for each register to have other address.

[0051] Accordingly, the CPU interface unit 14 sets each received interrupt signal tp-int, adf-int, pes-int, vid-int, aud-int and data-int at the point of time at which the interrupt is generated as of Fig. 17. In this time, the interrupt register and the interrupt enable value are capable of being read/written by the CPU 13.

[0052] The MPEG2 transport decoder operated as mentioned above is briefly described below.

[0053] Once the transport packet is inputted from the channel decoder through the data bus data3 to the transport parser unit 11, the transport packet decoder controller 21A is operated to parse the transport packet header, load each field value of the transport packet header to the corresponding register 21B, 23B and 24B and generates the interrupt to the CPU according to the interrupt enable state. The corresponding controller is operated by detecting if the required packet data is according to the state of PID-V-flag, PID-A-flag, PID-PAT-flag, PID-CAT-flag and PID-NIT-flag after the transport packet header is decoded. In case the ADF field exists in the packet, namely in case the checked bit of the ADF control is 10 or 11, the existing of the ADF field is detected , the ADF decoder controller 23A firstly and the corresponding controllers 22a and 24a are operated. The ADF decoder controller 23a loads each field value of the adf field to the corresponding register by parsing the adf field and generates the interrupt to the CPU 13 according to the interrupt enable condition. In case the ADF field decoding is finished or the ADF field does not exist, the corresponding decoder controller is operated. That is, in case PID-V-flag, PID-A-flag and PID-D-flag are set as 1, the PES decoder controller 24A is operated to parse the PES packet header, load each field value to the corresponding register and generate the interrupt to the CPU 13 according to the interrupt enable state. And in accordance with the sorts of the video, audio and data decoders, the PES packet data is transmitted to the video decoder interface unit 41, the audio decoder interface unit 42 and the data decoder interface unit 43.

[0054] While, the CPU 13 examines which area causes the CPU 13 to generate the interrupt whenever the interrupt exists in each block by loading each interrupt register value and processes the programmed interrupt according to the condition of the interrupt. The CPU 13 can access several registers in the transport parser unit 11 through the CPU interface unit 14 and access the video, audio and data decoders. On the other hand, the decoder interface unit 15 performs as a controller with the video, audio and data decoders as to transmitting the PES packet data decoded from the PES decoder 24 and having the CPU 13 access so that helps the CPU 13 to access each decoders in accordance with the applications after the programmed contents of user.

[0055] The above-mentioned MPEG2 transport decoder embodying the present invention has effects as follows:

First, the decoding operation may be variously applied according to the programmed contents by user by composing the MPEG2 transport decoder with incorporating a hardwired logic and a CPU capable of being programmed;
Secondly, the application of the described embodiment is various by incorporating the high- speed hardwired logic with the low-speed CPU;
Thirdly, the described embodiment solves the problem of developing new transport decoder occurring when the decoder is composed only with the hardwired logic circuit, the application range is limited in the associated application contents and characteristics;
Fourthly, by embodying each decoder interface, the data of the video, audio and data decoded from the hardwired transport parser unit are outputted to the corresponding decoders simultaneously with the CPU may be programmed parses the decode with the time-sharing, thereby accessing the decoder; and
Fifthly, while the CPU accesses one of each decoder, the decoder which the CPU does not access can transmits the data stored the memory or the FIFO.

Claims

1. An MPEG transport decoder comprising: a channel decoder unit for outputting a signal received through a satellite or a cable tuned to receive transport packet data; a transport decoder for decoding the transport packet data; and video (5), audio (6), and data decoders (7) for decoding video, audio and data signals via the transport decoder, said transport decoder further comprising:

   a transport parser unit (11) for storing at least one syntax field value in at least one packet decoder register, said syntax field value obtained by parsing said transport packet data, for outputting said transport packet data, whereby

each packet of said transport packet data is identified using a packet identifier (PID) obtained from said received transport packet data, and for outputting an interrupt signal if a predefined syntax field value is stored within any one of said packet decoder registers;
a CPU interface unit (14) for providing an interface between said at least one packet decoder register of said transport parser unit (11) and each of said video (5), audio (6) or data (7) decoders, for outputting a signal selecting either the transport parser unit (11) or the video (5), audio (6) or data (7) decoders or a first memory unit (22) the selection being made by decoding an address, the CPU interface unit further comprising at least one interrupt register for receiving at least one interrupt signal from either of said transport parser unit (11) or said video (5), audio (6) or data (7) decoders;
a CPU (13) for reading the contents of said interrupt register after the interrupt signal has been inputted from either the transport parser unit (11), the video (5), audio (6) or data decoder (7), for detecting whether the interrupt signal is inputted from said transport parser unit (11) or from the video (5), audio (6) or data decoder (7), and arbitrating CPU (13) access to said transport parsing unit (11), video (5), audio (6) or data decoders (7) according to a program which is stored on a second memory unit (12), and which is arranged to determine the operations of said CPU (13); and
a decoder interface unit (15) for controlling the order of exchange of said transport packet data amongst said CPU (13), said transport parser unit (11), or said video (5), audio (6) or data decoders (7).

2. An MPEG2 transport decoder as defined in claim 1, wherein said transport parser unit (11) comprises:

a channel decoder interface unit (20) for inputting or outputting the transport packing data and a controlling signal to said channel decoder unit (1);
a transport decoder (21) for storing a syntax field value associated with a header parsed from the transport packet data section of said transport packet data and generating the interrupt signal which is output to said CPU interface unit (14) if a defined syntax field value is stored within said transport decoder (21);
the first memory unit (22) for storing program specific information (PSI) sections of the MPEG2 streams, and an adaptation_extension_data, a transport_private_data, a pes_extension_data and a DSM_trick_mode_data at addresses of the memory designated by the CPU (13);
an ADF decoder (23) for storing a syntax field value associated with a header parsed from the ADF data of said transport packet data and generating the interrupt signal which is output to said CPU interface unit (14) if a defined syntax field value is stored within said transport decoder (21);
a PES decoder (24) for storing a syntax field value associated with a header parsed from the PES section of said transport packet data and generating the interrupt signal which is output to said CPU interface unit (14) if a defined syntax field value is stored within said transport decoder (21).

3. An MPEG2 transport decoder as defined in claim 2, wherein said transport decoder (21) further comprises:

a transport packet decoder controller (21a) for parsing said transport packet data; and
a transport packet decoder register (21b) for storing at least one header field value parsed in said transport packet decoder controller (21a) whereby said transport packet decoder register (21b) can be accessed by said CPU (13).

4. An MPEG2 transport decoder as defined in claim 2, wherein said first memory unit (22) comprises:

a memory (22a) for storing the PSI sections of the MPEG2 streams and the adaptation_extension_data, the transport_private_data, the PES_extension_data and the DSM_trick_mode_data contained within the selected packet, and a memory controller (22b) having start addresses and end addresses for storing the PSI data sections of the MPEG2 streams, the adaptation _extension_data, the transport_private_data, the PES_extension_data and the DSM_trick_mode_data at designated addresses in the memory (22a) and generating write addresses by automatically changing the write address, said write address starting with a start address and finishing with an end address for storing the PSI data sections of the MPEG2 streams, the adaptation_extension_data, the transport_private_data, the PES_extension_data and the DSM_trick_mode_data, whereby the memory (22a) receives the PSI data sections of the MPEG2 streams, the adaptation_extension_data, the transport_private_data, the PES_extension_data and the DSM_trick_mode_data and a control signal for storing the received data from the memory controller (22b) and allows the CPU (13) access to the received data.

5. An MPEG2 transport decoder as defined in claim 2, wherein said ADF decoder (23) comprises:

an ADF decoder controller (23a) for parsing said ADF data of said transport packet data; and
an ADF decoder register (23b) for storing at least one header field value parsed by said ADF decoder controller (23a) whereby the said ADF decoder registers (23b) are accessed by said CPU (13).

6. An MPEG2 transport decoder as defined in claim 2, wherein said PES decoder (24) comprises:

a PES decoder controller (24a) for parsing said PES section of said transport packet data; and
a PES decoder register (24b) for storing at least one header field value parsed by said PES decoder controller whereby the said PES decoder registers (24b) are accessed by CPU (13).

7. An MPEG2 transport decoder as defined in claim 1, wherein said CPU interface unit (14) comprises:

a data buffer (36) for buffering the contents of said CPU data bus;
a CPU address decoder (31) for generating a selecting signal for selecting one of at least one packet decoder register within said transport parser unit (11) by decoding part of the address supplied by said CPU, a selecting signal enabling the CPU (13) to access the video, or audio or data decoders or the first memory unit (22);
a tp-CPU interface unit (32) for generating a controlling signal cntrl-dsp-td allowing said CPU (13) to access said transport packet decoder register (21b) by incorporating a controlling signal generated by the CPU (13) with an address signal and the selecting signal;
a memory interface unit (33) for generating a controlling signal cntrl-dsp-mem allowing said CPU (13) to access said memory (22a) by incorporating a controlling signal generated by CPU (13) with an address signal and the selecting signal;
an adf-CPU interface unit (34) for generating a controlling signal cntrl-dsp-adf for the CPU (13) to access said ADF decoder register (23b) by incorporating a controlling signal generated by the CPU 13 with an address signal and the selecting signal; and
a pes-CPU interface unit (35) for generating a controlling signal cntrl-dsp-pes allowing said CPU (13) to access said PES decoder register (24b) by incorporating a controlling signal generated by the CPU (13) with an address signal and the selecting signal.

8. An MPEG2 transport decoder as defined in claim 7, wherein said tp-CPU interface unit (32), memory interface unit (33) and adf-CPU interface unit (34) decode the address for each of transport packet decoder register (21b) and memory (22a) and ADF decoder register (23b) respectively.

9. An MPEG2 transport decoder as defined in claim 1, wherein said decoder interface unit comprises:

a video decoder interface unit (41) for controlling a controlling signal enabling the CPU 13 and the PES decoder 24 to access the video decoder;
an audio decoder interface unit (42) for controlling a controlling signal enabling the CPU 13 and the PES decoder 24 to access the audio decoder; and
a data decoder interface unit for controlling a controlling signal enabling the CPU 13 and the PES decoder 24 to access the data decoder jointly.

10. An MPEG2 transport decoder as defined in claim 9, wherein said video, audio and data decoder interface units each comprise:

a first data buffer (51) for receiving, storing momentarily, and then outputting said controlling signal and a corresponding data signal from said CPU (13);
a third memory unit for storing the data outputted from said transport parser unit (11) and then outputting firstly previously inputted said controlling signal and corresponding data signal and simultaneously outputting signals fifo-ef and fifo-ff indicating whether the third memory unit is full or not;
a second data buffer (53) for storing momentarily then outputting said controlling and corresponding data signal from said third memory unit;
an access controlling unit (54) for setting a token signal giving right of an access to said CPU (13) to said video, audio or data decoders according to whether the CPU (13) requires access to the video, audio or data decoders and the signals fifo-ef and fifo-ff output from said third memory unit; and
an interface controlling unit (55) for controlling access of CPU (13) to said video, audio or data decoders to read/write to the video, audio or data decoder enabling the present accessing job to be completed after receiving said token signal from said access controlling unit (54).

11. An MPEG2 transport decoder is defined in claim 9, wherein said video, audio and data decoder inter-

face units can be partially composed in accordance with the application.

12. An MPEG2 transport decoder as defined in claims 9 and 10, wherein said video, audio and data decoder interface units each have shared access to said third memory unit.

13. An MPEG2 transport decoder as defined in claim 12, wherein said third memory unit comprises:

a third data buffer (61) for storing momentarily the output data of said transport parser unit (11);
a memory being divided into three areas of video, audio and data storing areas and storing the data inputted from said third data buffer (61) or outputting the data previously stored from either the video, audio or data decoder;
a video writing pointer (62) for outputting a writing address in order to write the video data to said video storing area;
an audio writing pointer (63) for outputting a writing address in order to write the audio data to said audio storing area;
a data writing pointer (64) for outputting a writing address in order to write the data to said data storing area;
a first address buffer (65) for storing momentarily the address outputted from said video writing pointer (62);
a second address buffer (66) for storing momentarily the address outputted from said audio writing pointer (63);
a third address buffer (67) for storing momentarily the address outputted from said data writing pointer (64);
a video reading pointer (72) for outputting a reading address in order to read the video data from said video storing area;
an audio reading pointer (73) for outputting a reading address in order to read the audio data from said audio storing area;
a data reading pointer (74) for outputting a reading address in order to read the data from said data storing area;
a fourth address buffer (69) for storing momentarily the address outputted from said video reading pointer (72);
a fifth address buffer (70) for storing momentarily the address outputted from said audio reading pointer (73);
a sixth address buffer (71) for storing momentarily the address outputted from said data reading pointer (74); and
a memory interface controller (75) for controlling the operations of said video, audio and data reading/writing pointers and said first to sixth address buffers by the controlling signal cntrl5 from the PES decoder (24).

Patentansprüche

1. MPEG-Transport-Dekoder, umfassend:

eine Kanal-Dekoder-Einheit zur Ausgabe eines Signals, welches durch einen Satelliten oder durch ein auf Empfang von Transport-Paket-Daten abgestimmtes Kabel empfangen wurde;

einen Transport-Dekoder zum Dekodieren der Transport-Paket-Daten;

ein Video- (5), Audio- (6), und Daten-Dekoder (7) zum Dekodieren von Video-, Audio- und Daten-Signalen mittels des Transport-Dekoders, wobei der Transport-Dekoder weiterhin umfaßt:

eine Transport-Parser-Einheit (11) zum Speichern von wenigstens einem Syntaxfeldwert in wenigstens einem Paket-Dekoder-Register, wobei der Syntaxfeldwert durch Parsen der Transport-Paket-Daten erhalten wurde, zur Ausgabe der Transport-Paket-Daten, wobei jedes Paket der Transport-Paket-Daten unter Verwendung eines Paket-Identifizierers (PID) identifiziert wird, welcher aus den empfangenen Transport-Paket-Daten erhalten wurde, und zur Ausgabe eines Interrupt-Signals, wenn ein vorbestimmter Syntaxfeldwert innerhalb eines der Paket-Dekoder-Register gespeichert ist;

eine CPU Schnittstelleneinheit (14) zum Bereitstellen einer Schnittstelle zwischen dem wenigstens einen Paket-Dekoder-Register der Transport-Parser-Einheit (11) und jedem der Video- (5), Audio- (6) oder Daten-Dekoder (7), zur Ausgabe eines Signales, wobei entweder die Transport-Parser-Einheit (11), oder der Video- (5), Audio- (6) oder Daten-Dekoder (7) oder eine erste Speichereinheit (22) gewählt wird, wobei die Auswahl durch Dekodieren einer Adresse getroffen wird, und die CPU Schnittstelle weiterhin wenigstens ein Interrupt-Register zum Empfangen von wenigstens einem Interrupt-Signal entweder von der Transport-Parser-Einheit (11) oder den Video- (5), den Audio- (6) oder den Daten-Dekodern (7) umfasst;

eine CPU (13) zum Lesen des Inhalts des

Interrupt-Registers, nachdem das Interrupt-Signal entweder von der Transport-Parser-Einheit (11), dem Video- (5), dem Audio-(6) oder dem Daten-Dekoder (7) eingegeben wurde, zum Erfassen, ob das Interrupt-Signal von der Transport-Parser-Einheit (11) oder von dem Video- (5), Audio- (6) oder Daten-Dekoder (7) eingegeben wurde, und zur Entscheidung über einen CPU-Zugang (13) zu der Transport-Parser-Einheit (11), dem Video- (5), dem Audio- (6) oder dem Daten-Dekoder (7) gemäß einem Programm, welches in einer zweiten Speichereinheit (12) gespeichert ist, und welche dazu ausgebildet ist, die Operationen der CPU (13) zu bestimmen; und

eine Dekoder-Schnittstellen-Einheit (15) zum Steuern der Reihenfolge des Austausches der Transport-Paket-Daten zwischen der CPU (13), der Transport-Parser-Einheit (11), oder der Video- (5), der Audio- (6) oder der Daten-Dekoder (7).

2. MPEG2-Transport-Dekoder nach Anspruch 1, worin die Transport-Parser-Einheit (11) folgendes umfaßt:

eine Kanal-Dekoder-Schnittstellen-Einheit (20) zur Eingabe oder Ausgabe der Transport-Paket-Daten und eines SteuerSignals zu der Kanal-Dekoder-Einheit (1);

einen Transport-Dekoder (21) zum Speichern eines Syntax-Feldwertes, welcher einer Kopfzeile zugeordnet ist, welche aus dem Transport-Paket-Daten-Bereich der Transport-Paket-Daten geparst wurde, und zum Erzeugen des Interrupt-Signals, welches zu der CPU-Schnittstellen-Einheit (14) ausgegeben wird, wenn ein definierter Syntax-Feldwert in dem Transport-Dekoder (21) gespeichert ist;

die erste Speichereinheit (22) zum Speichern von Bereichen an programm-spezifischer Information (PSI) der MPEG2-Ströme, und Adaptation_extension_data, Transport _private_data, Pes_extension_data und DSM_trick_mode_data in von der CPU (13) bestimmten Adressen des Speichers;

einen ADF-Dekoder (23) zum Speichern eines Syntax-Feldwertes, welcher einer Kopfzeile zugeordnet ist, welche aus den ADF-Daten der Transport-Paket-Daten geparst wurde, und zum Erzeugen des Interrupt-Signals, welches zu der CPU Schnittstellen-Einheit (14) ausgegeben wird, wenn ein definierter Syntax-Feldwert in dem Transport-Dekoder (21) gespeichert ist;

einen PES-Dekoder (24) zum Speichern eines Syntax-Feldwertes, welcher einer Kopfzeile zugeordnet ist, welche aus den PES-Daten der Transport-Paket-Daten geparst wurde, und zum Erzeugen des Interrupt-Signals, welches zu der CPU Schnittstellen-Einheit (14) ausgegeben wird, wenn ein definierter Syntax-Feldwert in dem Transport-Dekoder (21) gespeichert ist.

3. MPEG2-Transport-Dekoder nach Anspruch 2, worin der Transport-Dekoder (21) weiterhin umfaßt:

einen Transport-Paket-Dekoder-Regler (21a) zum Parsen der Transport-Paket-Daten; und

ein Transport-Paket-Dekoder-Register (21b) zum Speichern von wenigstens einem Kopf-Feldwert, welcher in dem Transport-Paket-Dekoder-Regler (21a) geparst wurde, wobei auf das Transport-Paket-Dekoder-Register (21b) von der CPU (13) zugegriffen werden kann.

4. MPEG2-Transport-Dekoder nach Anspruch 2, worin die erste Speichereinheit (22) umfaßt;

einen Speicher (22a) zum Speichern der PSI-Bereiche der MPEG2-Ströme und der Adaptation_extension_data, der Transport_ private_data, der Pes_extension_data und der DSM_trick_mode_data, welche in dem ausgewählten Paket enthalten sind, und eine Speicher-Steuerung (22b), welche Start-Adressen und End-Adressen zum Speichern der PSI-Daten-Bereiche der MPEG2-Ströme. der Adaptation_extension_data, der Transport_ private_data, der Pes_extension_data und der DSM_trick_mode_data in bestimmten Adressen im Speicher (22a) hat, und zum Erzeugen von Schreib-Adressen durch automatisches Ändern der Schreib-Adressen, wobei die Schreib-Adresse mit einer Start-Adresse beginnt und mit einer End-Adresse endet zum Speichern der PSI-Daten-Bereiche der MPEG2-Ströme, der Adaptation_extension_ data, der Transport_private_data, der Pes_ extension_data und der DSM_trick_ mode_data, wobei der Speicher (22a) die PSI-Daten-Bereiche der MPEG2-Ströme empfängt, die Adaptation_extension_data, die Transport _private data, die Pes_extension_ data und die DSM_trick_mode_data und ein Steuersignal zum Speichern der empfangenen Daten aus der Speicher-Steuerung (22b) und

der CPU (13) Zugriff auf die empfangenen Daten gestattet.

5. MPeG2-Transport-Dekoder nach Anspruch 2, worin der ADF-Dekoder (23) umfaßt:

   einen ADF-Dekoder-Regler (23a) zum Parsen der ADF-Daten der Transport-Paket-Daten; und

   ein ADF-Dekoder-Register (23b) zum Speichern von wenigstens einem Kopf-Feldwert, welcher von der ADF-Dekoder-Steuerung (23b) geparst wurde, wobei auf die ADF-Dekoder-Register (23b) von der CPU (13) zugegriffen wird.

6. MPEG2-Transport-Dekoder nach Anspruch 2, worin der PES-Dekoder (24) umfaßt:

   eine PES-Dekoder-Steuerung (24a) zum Parsen des PES-Bereiches der Transport-Paket-Daten; und

   ein PES-Dekoder-Register (24b) zum Speichern von wenigstens einem Kopf-Feldwert, welcher von der PES-Dekoder-Steuerung geparst wurde, wobei auf die PES-Dekoder-Register (24b) von der CPU (13) zugegriffen wird.

7. MPEG2-Transport-Dekoder nach Anspruch 1, worin die CPU-Schnittstellen-Einheit (14) folgendes umfaßt:

   einen Daten-Puffer (36) zum Puffern des Inhalts des CPU-Datenbusses;

   einen CPU-Adressen-Dekoder (31) zum Erzeugen eines Auswahlsignals zum Auswählen eines von wenigstens einem Paket-Dekoder-Register innerhalb der Transport-Parser-Einheit (11) durch Dekodieren eines Teils der Adresse, welche von der CPU zur Verfügung gestellt wurde, wobei ein Auswahlsignal der CPU (13) ermöglicht, auf die Video-, die Audio- oder die Daten-Dekoder oder die erste Speichereinheit (22) zuzugreifen;

   eine tp-CPU Schnittstellen-Einheit (32) zum Erzeugen eines Steuersignals cntrl-dsp-td, welches der CPU (13) gestattet auf das Transport-Paket-Dekoder-Register (21b) durch Einbeziehung eines Steuersignals zuzugreifen, welches von der CPU (13) mit einem Adressen-Signal und dem Auswahl-Signal erzeugt wurde;

   eine Speicher-Schnittstellen-Einheit (33) zum Erzeugen eines Steuersignals cntrl-dsp-mem, welches der CPU (13) gestattet auf den Speicher (22a) durch Einbeziehung eines Steuersignals zuzugreifen, welches von der CPU (13) mit einem Adressen-Signal und dem Auswahl-Signal erzeugt wurde;

   eine adf-CPU-Schnittstellen-Einheit (34) zum Erzeugen eines Steuersignals cntrl-dsp-adf, welches der CPU (13) gestattet auf das ADF-Dekoder-Register (23b) durch Einbeziehung eines Steuersignals zuzugreifen, welches von der CPU (13) mit einem Adressen-Signal und dem Auswahl-Signal erzeugt wurde; und

   eine pes-CPU-Schnittstellen-Einheit (35) zum Erzeugen eines Steuersignals cntrl-dsp-pes, welches der CPU (13) gestattet auf das PES-Dekoder-Register (24b) durch Einbeziehung eines Steuersignals zuzugreifen, welches von der CPU (13) mit einem Adressen-Signal und dem Auswahl-Signal erzeugt wurde.

8. MPEG2-Transport-Dekoder nach Anspruch 7, worin die tp-CPU-Schnittstellen-Einheit (32), Speicher-Schnittstellen-Einheit (33) und adf-CPU-Schnittstellen-Einheit (34) die Adresse für jedes Transport-Paket-Dekoder-Register (21b) und Speicher (22a) und ADF-Dekoder-Register (23b) beziehungsweise dekodieren.

9. MPEG2-Transport-Dekoder nach Anspruch 1, worin die Dekoder-Schnittstellen-Einheit umfaßt:

   eine Video-Dekoder-Schnittstellen-Einheit (41) zum Steuern eines Steuersignals, welches die CPU (13) und den PES-Dekoder (24) befähigt, auf den Video-Dekoder zuzugreifen;

   eine Audio-Dekoder-Schnittstellen-Einheit (42) zum Steuern eines Steuersignals, welches die CPU (13) und den PES-Dekoder (24) befähigt, auf den Audio-Dekoder zuzugreifen; und

   eine Daten-Dekoder-Schnittstellen-Einheit zum Steuern eines Steuersignals, welches die CPU (13) und den PES-Dekoder (24) befähigt, auf den Daten-Dekoder gemeinsam zuzugreifen.

10. MPEG2-Transport-Dekoder nach Anspruch 9, worin die Video-, Audio- und Daten-Dekoder-Schnittstellen-Einheiten jeweils umfassen:

   einen ersten Daten-Puffer (51) zum Empfangen, momentanen Speichern, und dann Ausgeben des Steuersignals und eines korrespondierenden Datensignals aus der CPU (13);

eine dritte Speichereinheit zum Speichern der Daten, welche von der Transport-Parser-Einheit (11) ausgegeben wurden, und dann zum Ausgeben des vorangehend zuerst eingegebenen Steuersignals und des korrespondierenden Datensignals und zum gleichzeitigen Ausgeben der Signale fifo-ef und fifo-ff, welche angeben, ob die dritte Speichereinheit voll ist oder nicht;

einen zweiten Daten-Puffer (53) zum momentanen Speichern und dann Ausgeben des Steuer- und korrespondierenden Datensignals aus der dritten Speichereinheit;

eine Zugangs-Steuer-Einheit (54) zum Setzen eines Kennungs-Signals, welches der CPU (13) Zugangsberechtigung zu den Video-, den Audio- oder den Daten-Dekodern gibt, je nach dem, ob die CPU (13) Zugang zu den Video-, den Audio- oder den Daten-Dekodern und den von der dritten Speichereinheit ausgegebenen Signalen fifo-ef und fifo-ff erfordert; und

eine Schnittstellen-Steuer-Einheit (55) zum Steuern des Zugangs der CPU (13) zu dem Video-, Audio- oder Daten-Dekoder, um den Video-, Audio- oder Daten-Dekoder zu lesen/beschreiben, wobei der gegenwärtige Zugriffsjob befähigt wird, nach Empfang des Kennungs-Signals von der Zugangssteuereinheit (54) ausgeführt zu werden.

11. MPEG2-Transport-Dekoder nach Anspruch 9, worin die Video-, Audio- und Daten-Dekoder-Schnittstellen-Einheiten teilweise in Übereinstimmung mit der Anwendung zusammengesetzt sein können.

12. MPEG2-Transport-Dekoder nach Anspruch 9, worin die Video-, Audio- und Daten-Dekoder-Schnittstellen-Einheiten jeweils gemeinsamen Zugang zu der dritten Speichereinheit haben.

13. MPEG2-Transport-Dekoder nach Anspruch 12, worin die dritte Speichereinheit umfaßt:

einen dritten Daten-Puffer (61) zum momentanen Speichern der Ausgabe-Daten der Transport-Parser-Einheit (11);

einen Speicher, der in drei Abschnitte von Video-, Audio- und Datenspeicherabschnitten eingeteilt ist und der die von dem dritten Daten-Puffer (61) eingegebenen Daten speichert oder die zuvor entweder von dem Video-, Audio- oder Daten-Dekoder gespeicherten Daten ausgibt;

einen Video-Schreibe-Zeiger (62) zum Ausgeben einer Schreib-Adresse, um die Video-Daten zu den Video-Speicher-Abschnitten zu schreiben;

einen Audio-Schreibe-Zeiger (63) zum Ausgeben einer Schreib-Adresse, um die Audio-Daten zu den Audio-Speicher-Abschnitten zu schreiben;

einen Daten-Schreibe-Zeiger (64) zum Ausgeben einer Schreib-Adresse, um die Daten zu den Daten-Speicher-Abschnitten zu schreiben;

einen ersten Adress-Puffer (65) zum momentanen Speichern der von dem Video-Schreibe-Zeiger (62) ausgegebenen Adresse;

einen zweiten Adress-Puffer (66) zum momentanen Speichern der von dem Audio-Schreibe-Zeiger (63) ausgegebenen Adresse;

einen dritten Adress-Puffer (67) zum momentanen Speichern der von dem Daten-Schreibe-Zeiger (64) ausgegebenen Adresse;

einen Video-Lese-Zeiger (72) zum Ausgeben einer Lese-Adresse, um die Video-Daten aus dem Video-Speicher-Abschnitt auszulesen;

einen Audio-Lese-Zeiger (73) zum Ausgeben einer Lese-Adresse, um die Audio-Daten aus dem Audio-Speicher-Abschnitt auszulesen;

einen Daten-Lese-Zeiger (74) zum Ausgeben einer Lese-Adresse, um die Daten aus dem Daten-Speicher-Abschnitt auszulesen;

einen vierten Adress-Puffer (69) zum momentanen Speichern der von dem Video-Lese-Zeiger (72) ausgegebenen Adresse;

einen fünften Adress-Puffer (70) zum momentanen Speichern der von dem Audio-Lese-Zeiger (73) ausgegebenen Adresse;

einen sechsten Adress-Puffer (71) zum momentanen Speichern der von dem Daten-Lese-Zeiger (72) ausgegebenen Adresse; und

eine Speicher-Schnittstellen-Steuerung (75) zum Steuern der Operationen der Video-, Audio- und Daten-Lese/Schreib-Zeiger und des ersten bis sechsten Adress-Puffers durch das Steuersignal cntrl5 von dem PES-Dekoder (24).

Revendications

1. Décodeur de transport MPEG comprenant : une unité de décodage de voie pour transmettre un signal reçu via un satellite ou un câble réglé ou synthonisé pour recevoir des données de paquets de transport ; un décodeur de transport pour décoder les données de paquets de transport ; et des décodeurs vidéo (5), audio (6) et de données (7) pour décoder les signaux vidéo, audio et de données via le décodeur de transport, ledit décodeur de transport comprenant en outre :

   une unité d'analyse de transport (11) pour stocker au moins une valeur de champ syntaxique dans au moins un registre décodeur de paquet, ladite valeur de champ syntaxique obtenue en analysant lesdites données de paquets de transport, pour transmettre lesdites données de paquets de transport, moyennant quoi chaque paquet desdites données de paquets de transport est identifié en utilisant un identificateur de paquet (PID) obtenu à partir desdites données de paquets de transport reçues, et pour transmettre un signal d'interruption si une valeur de champ syntaxique prédéterminée est stockée dans l'un desdits registres décodeurs de paquets.
   une unité d'interface UC (14) pour assurer une interface entre ledit au moins un registre décodeur de paquets de ladite unité d'analyse de transport (11) et chacun desdits décodeurs vidéo (5), audio (6) et de données (7), pour transmettre un signal sélectionnant l'unité d'analyse de transport (11) ou bien les décodeurs vidéo (5), audio (6) ou de données (7) ou une première mémoire (22), la sélection étant effectuée en décodant une adresse, l'unité d'interface UC comprenant au moins un registre d'interruption pour recevoir au moins un signal d'interruption provenant de l'unité d'analyse de transport (11) ou bien desdits décodeurs vidéo (5), audio (6) et de données (7) ;
   une UC (13) pour lire le contenu dudit registre d'interruption une fois le signal d'interruption entré à partir de l'unité d'analyse de transport (11), du décodeur vidéo (5), audio (6) ou de données (7), pour détecter si le signal d'interruption est entré à partir de ladite unité d'analyse de transport (11) ou du décodeur vidéo (5), audio (6) ou de données (7), et pour arbitrer l'accès de l'UC (13) à ladite unité d'analyse de transport (11), auxdits décodeurs vidéo (5), audio (6) ou de données (7) selon un programme qui est enregistré dans une seconde mémoire (12), et qui est conçu pour déterminer les opérations de ladite UC (13) ; et
   une unité d'interface décodeur (15) pour contrôler l'ordre d'échange desdites données de paquets de transport entre ladite UC (13), ladite unité d'analyse de transport (11) ou ledit décodeur vidéo (5), audio (6) ou de données (7).

2. Décodeur de transport MPEG2 selon la revendication 1, dans lequel ladite unité d'analyse de transport (11) comprend :

   une unité d'interface décodeur de voie (20) pour la réception/transmission de données de paquets de transport et d'un signal de contrôle à ladite unité de décodeur de voie (1) ;
   un décodeur de transport (21) pour stocker une valeur de champ syntaxique associée à un en-tête analysé à partir de la section des données de paquets de transport desdites données de paquets de transport et générer le signal d'interruption qui est transmis à l'unité d'interface UC (14) si une valeur de champ syntaxique définie est stockée dans ledit décodeur de transport (21) ;
   la première unité de mémoire (22) pour stocker les sections d'informations spécifiques au programme (PSI) des flux MPEG2 et des données adaptation_extension_data, transpon-private-data, pes_extension_data et DMS_trick_ mode_data aux adresses de la mémoire désignée par l'UC (13) ;
   un décodeur ADF (23) pour stocker une valeur de champ syntaxique associée à un en-tête analysé à partir des données ADF desdites données de paquets de transport et générer le signal d'interruption qui est transmis à ladite unité d'interface UC (14) si une valeur de champ syntaxique est stockée dans ledit décodeur de transport (21);
   un décodeur PES (24) pour stocker une valeur de champ syntaxique associée à un en-tête analysé à partir de la section PES desdites données de paquets de transport et générer le signal d'interruption qui est transmis à ladite unité d'interface UC (14) si une valeur de champ syntaxique est stockée dans ledit décodeur de transport (21).

3. Décodeur de transport MPEG2 selon la revendication 2, dans lequel ledit décodeur de transport (21) comprend en outre :

   un contrôleur de décodeur de paquets de transport (21a) pour analyser lesdites données de paquets de transport ; et
   un registre de décodeur de paquets de transport (21b) pour stocker au moins une valeur de champ d'en-tête analysée dans ledit contrôleur de décodeur de paquets de transport (21a), moyennant quoi l'UC (13) peut accéder audit

registre décodeur de paquets de transport (21b).

4. Décodeur de transport MPEG2 selon la revendication 2, dans lequel ladite première unité de mémoire (22) comprend :

une mémoire (22a) pour stocker les sections PSI des flux MPEG2 et des données adaptation_extension_data, transport_private_data, PES_extension_data, et DSM_trick_mode_data contenues dans le paquet sélectionné, et un contrôleur de mémoire (22b) ayant des adresses de début et des adresses de fin pour stocker les sections de données PSI des flux de MPEG2, les données adaptation_extension_data, transport_private_data, PES_extension_data et DSM_trick_mode_data aux adresses désignées dans la mémoire (22a) et générer des adresses d'écriture en changeant automatiquement l'adresse d'écriture, ladite adresse d'écriture commençant par une adresse de début et finissant par une adresse de fin pour stocker les sections de données PSI des flux de MPEG2, les données adaptation_extension_data, transport_private_data, PES_extension_data et DSM_trick_mode_data, moyennant quoi la mémoire (22a) reçoit les sections de données PSI des flux de MPEG2, les données adaptation_extension_data, transport_private_data, PES_extension_data et DSM_trick_mode_data et un signal de contrôle pour stocker les données reçues à partir du contrôleur de mémoire (22b) et permet l'accès de l'UC (13) aux données reçues,

5. Décodeur de transport MPEG2 selon la revendication 2, dans lequel ledit décodeur ADF (23) comprend :

un contrôleur de décodeur ADF (23a) pour analyser lesdites données ADF desdites données de paquets de transport ; et
un registre décodeur ADF (23b) pour stocker au moins une valeur de champ d'en-tête analysée par ledit registre décodeur ADF (23a) moyennant quoi ladite UC (13) accède auxdits registres décodeurs ADF (23b).

6. Décodeur de transport MPEG2 selon la revendication 2, dans lequel ledit décodeur PES (24) comprend :

un contrôleur décodeur PES (24a) pour analyser ladite section PES desdites données de paquets de transport ; et
un registre décodeur PES (24b) pour stocker au moins une valeur de champ d'en-tète analysée par ledit contrôleur décodeur PES moyennant quoi l'UC (13) accède auxdits registres décodeurs PES (24b).

7. Décodeur de transport MPEG2 selon la revendication 1, dans lequel ladite unité d'interface UC (14) comprend :

un tampon de données (36) pour réaliser un tamponnage des contenus desdits bus de données UC ;
un décodeur d'adresse UC (31) pour générer un signal de sélection pour sélectionner l'un parmi au moins un registre décodeur de paquets dans ladite unité d'analyse de transport (11) en décodant une partie de l'adresse fournie par ladite UC, un signal de sélection permettant à l'UC (13) d'accéder aux décodeurs vidéo, audio, et de données ou à la première unité de mémoire (22) ;
une unité d'interface tp-UC (32) pour générer un signal de contrôle cntrl-dsp-td permettant à ladite UC (13) d'accéder audit registre décodeur de paquets de transport (21b)en incorporant un signal de contrôle généré par l'UC (13) avec un signal d'adresse et le signal sélectif ;
une unité d'interface de mémoire (33) pour générer un signal de contrôle cntrl-dsp-mem permettant à ladite UC (13) d'accéder à ladite mémoire (22a) en incorporant un signal de contrôle généré par l'UC (13) avec un signal d'adresse et le signal de sélection ;
une unité d'interface adf-UC (34) pour générer un signal de contrôle cntrl-dsp-adf pour que l'UC (13) accède audit registre décodeur ADF (23b) en incorporant un signal de contrôle généré par l'UC (13) avec un signal d'adresse et le signal de sélection ; et
une unité d'interface pes-UC (35) pour générer un signal de contrôle cntrl-dsp-pes permettant à ladite UC (13) d'accéder audit registre décodeur PES (24b) en incorporant un signal de contrôle généré par l'UC (13) avec un signal d'adresse et le signal de sélection.

8. Décodeur de transport MPEG2 selon la revendication 7, dans lequel lesdites unité d'interface tp-UC (32), unité d'interface mémoire (33) et unité d'interface adf-UC (34) décodent l'adresse pour chacun des éléments suivants, respectivement : le registre décodeur de paquets de transport (21b), la mémoire (22a) et le registre décodeur ADF (23b).

9. Décodeur de transport MPEG2 selon la revendication 1, dans lequel ladite unité d'interface décodeur comprend :

une unité d'interface décodeur vidéo (41) pour contrôler un signal de contrôle permettant à l'UC 13 et au décodeur PES 24 d'accéder au décodeur vidéo ;
une unité d'interface décodeur audio (42) pour contrôler un signal de contrôle permettant à l'UC 13 et au décodeur PES 24 d'accéder au décodeur audio ; et
une unité d'interface de décodeur de données pour contrôler un signal de contrôle permettant à l'UC 13 et au décodeur PES 24 d'accéder conjointement au décodeur de données.

10. Décodeur de transport MPEG2 selon la revendication 9, dans lequel lesdites unités d'interface de décodeur vidéo, audio et de données comprennent chacune :

un premier tampon de données (51) pour recevoir, stocker provisoirement, et ensuite transmettre ledit signal de contrôle et un signal de données correspondant provenant de ladite UC (13) ;
une troisième unité de mémoire pour stocker les données transmises par ladite unité d'analyse de transport (11) et transmettant ensuite tout d'abord ledit signal de contrôle et ledit signal de données correspondant transmis précédemment et transmettant simultanément les signaux fifo-ef et fifo-ff indiquant si la troisième unité mémoire est pleine ou non ;
un second tampon de données (53) de stockage provisoire transmettant ensuite ledit signal de contrôle et de données correspondant provenant de ladite troisième unité de mémoire ;
une unité de contrôle d'accès (54) pour déterminer un signal de jeton permettant à ladite UC (13) d'accéder auxdits décodeurs vidéo, audio et de données selon que l'UC (13) nécessite l'accès aux décodeurs vidéo, audio et de données et les signaux fifo-ef et fifo-ff transmis par ladite troisième unité de mémoire ; et
une unité de contrôle interface (55) pour contrôler l'accès de l'UC (13) auxdits décodeurs vidéo, audio ou de données pour une lecture/écriture au décodeur vidéo, audio et de données permettant que l'actuel travail d'accès soit terminé après réception dudit signal de jeton en provenance de ladite unité de contrôle d'accès (54).

11. Décodeur de transport MPEG2 selon la revendication 9, dans lequel lesdites unités d'interface décodeur vidéo, audio et de données peuvent être partiellement constituées conformément à l'application.

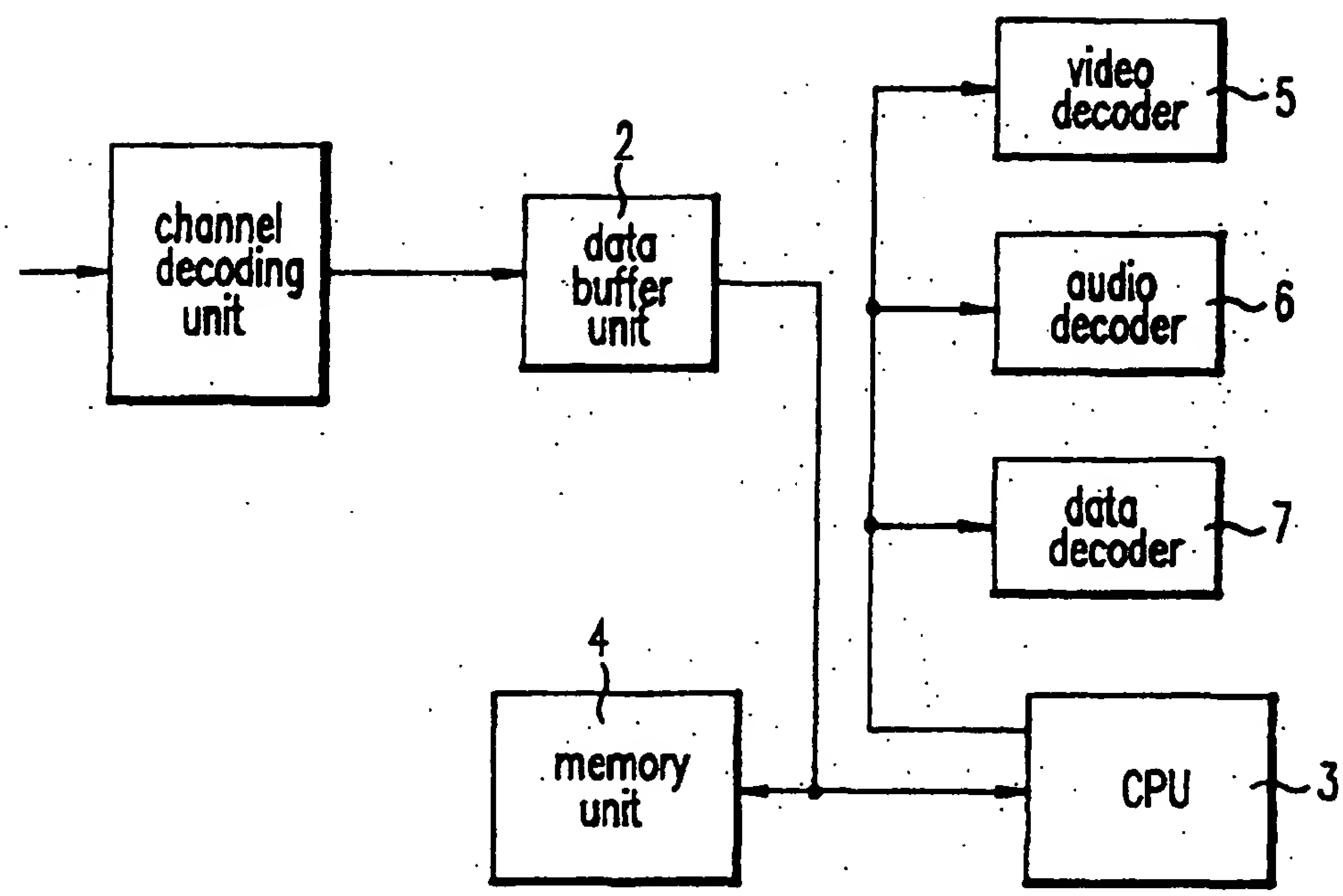
12. Décodeur de transport MPEG2 selon les revendications 9 et 10, dans lequel lesdites unités d'interface décodeur vidéo, audio, et de données ont chacune un accès partagé à ladite troisième unité de mémoire.

13. Décodeur de transport MPEG2 selon la revendication 12, dans lequel ladite troisième unité de mémoire comprend:

un troisième tampon de données (61) pour stocker provisoirement les données transmises de ladite unité d'analyse de transport (11);
une mémoire divisée en trois zones de stockage vidéo, audio et de données et stockant les données reçues dudit troisième tampon de données (61) ou transmettant les données précédemment stockées par le décodeur vidéo, audio ou de données ;
un pointeur d'écriture vidéo (62) pour transmettre une adresse d'écriture afin d'écrire les données vidéo à ladite zone de stockage vidéo ;
un pointeur d'écriture audio (63) pour transmettre une adresse d'écriture afin d'écrire les données audio à ladite zone de stockage audio ;
un pointeur d'écriture de données (64) pour transmettre une adresse d'écriture afin d'écrire les données à ladite zone de stockage ;
un premier tampon d'adresse (65) pour stocker provisoirement l'adresse transmise par ledit pointeur d'écriture vidéo (62) ;
un second tampon d'adresse (66) pour stocker provisoirement l'adresse transmise par ledit pointeur d'écriture audio (63) ;
un troisième tampon d'adresse (67) pour stocker provisoirement l'adresse transmise par ledit pointeur d'écriture de données (64) ;
un pointeur de lecture vidéo (72) pour transmettre une adresse de lecture afin de lire les données vidéo provenant de ladite zone de stockage vidéo ;
un pointeur de lecture audio (73) pour transmettre une adresse de lecture afin de lire les données audio provenant de ladite zone de stockage audio ;
un pointeur de lecture de données (74) pour transmettre une adresse de lecture afin de lire les données provenant de ladite zone de stockage de données ;
un quatrième tampon d'adresse (69) pour stocker provisoirement l'adresse transmise par ledit pointeur de lecture vidéo (72) ;
un cinquième tampon d'adresse (70) pour stocker provisoirement l'adresse transmise par ledit pointeur de lecture audio (73) ;
un sixième tampon d'adresse (71) pour stocker provisoirement l'adresse transmise par ledit pointeur de lecture de données (74) ; et
un contrôleur d'interface mémoire (75) pour

contrôler les opérations desdits pointeurs de lecture/écriture vidéo, audio et de données et desdits premier au sixième tampons d'adresse par le signal de contrôle cntrl5 provenant du décodeur PES (24).

F I G.1
prior art



F I G.2
prior art

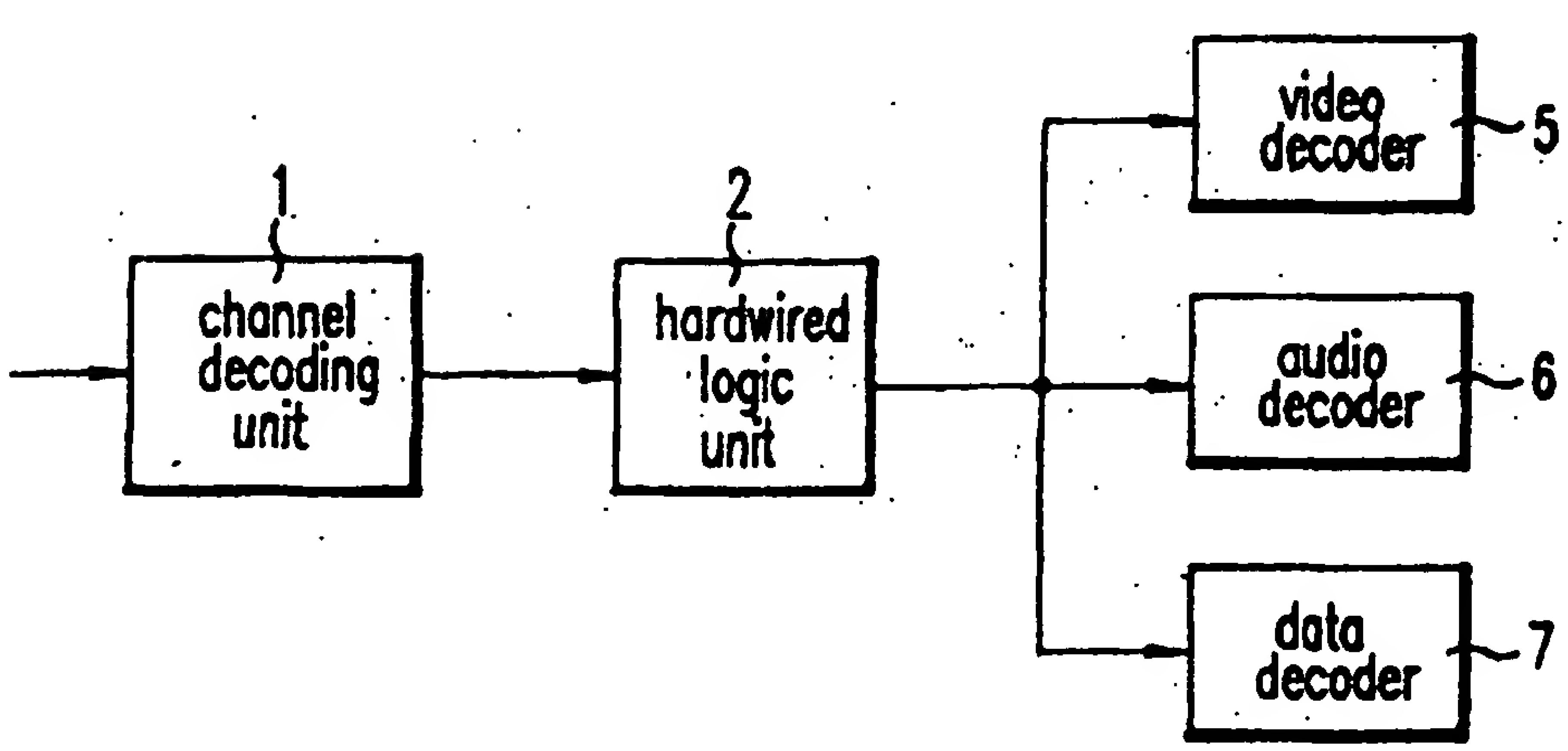
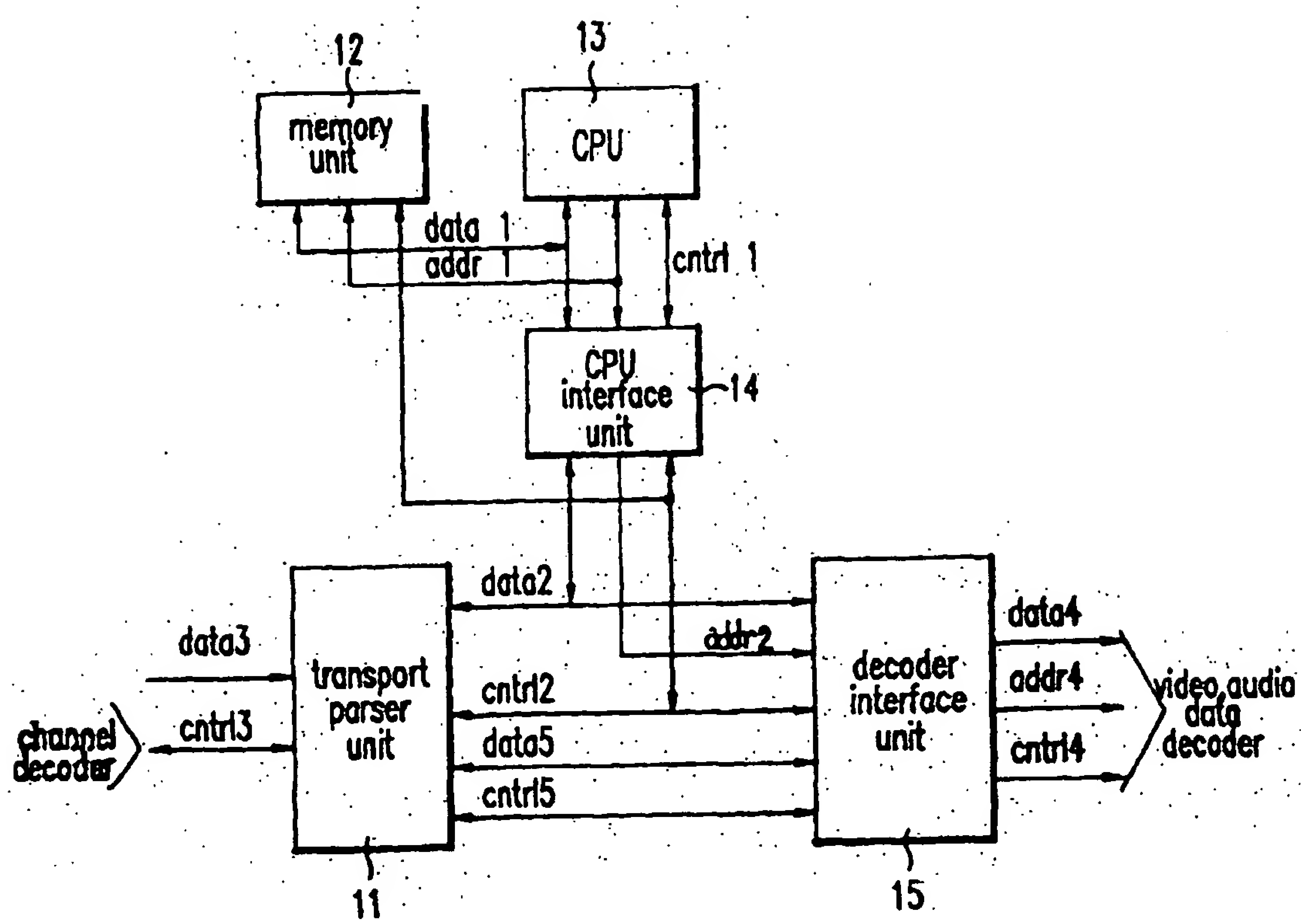


FIG.3


12
memory unit
13
CPU
data 1
addr 1
cntrl 1
CPU interface unit
14
data2
addr2
data3
transport parser unit
cntrl2
channel decoder
cntrl3
data5
cntrl5
11
decoder interface unit
15
data4
addr4
cntrl4
video,audio data decoder

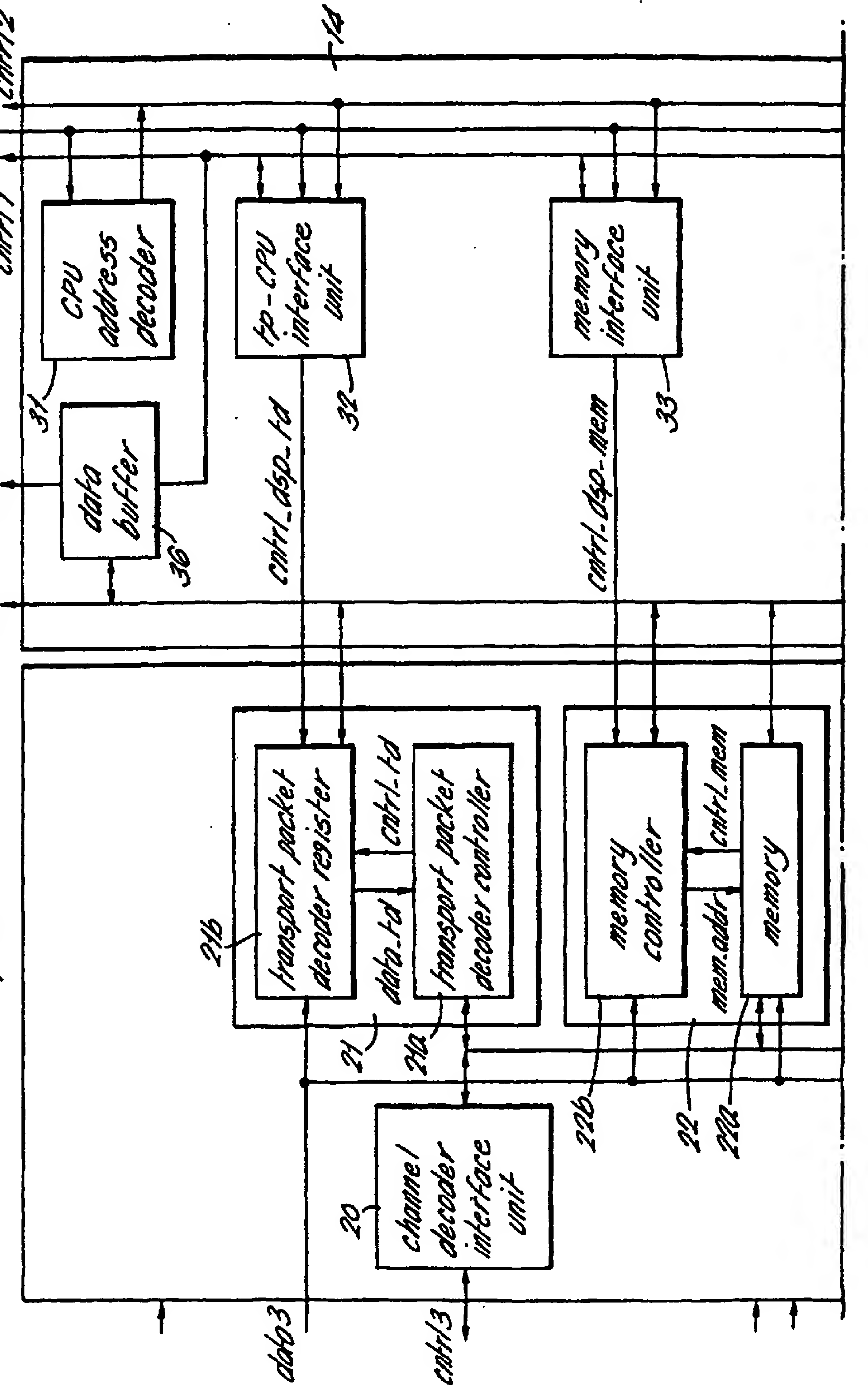

cntrl2
14
addr1
cntrl1
CPU address decoder
31
tp-CPU interface unit
32
memory interface unit
33
data buffer
36
data1
data2
cntrl_dsp_td
cntrl_dsp_mem
transport packet decoder register
transport packet decoder controller
cntrl_td
data_td
21b
21a
21
memory controller
memory
cntrl_mem
mem_addr
22b
22
22a
channel decoder interface unit
20
data3
cntrl3

FIG. 4 (a).

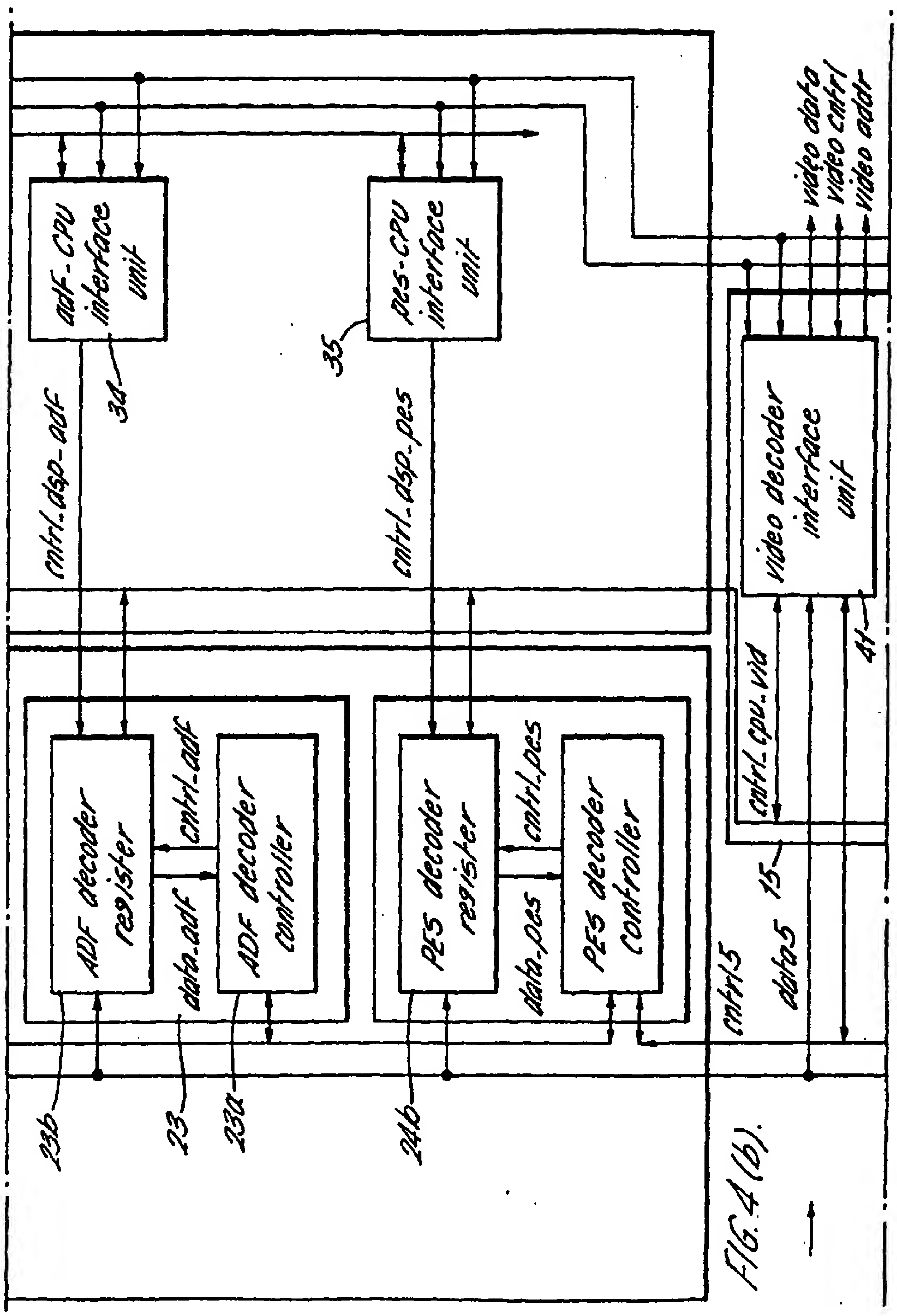

adf-CPU interface unit
34
pes-CPU interface unit
35
cntrl.dsp-adf
cntrl.dsp-pes
video data
video cntrl
video addr
video decoder interface unit
41
cntrl.cpu.vid
15
data5
cntrl5
ADF decoder register
cntrl.adf
data.adf
ADF decoder controller
PES decoder register
cntrl.pes
data.pes
PES decoder controller
23b
23
23a
24b

FIG.4(b).

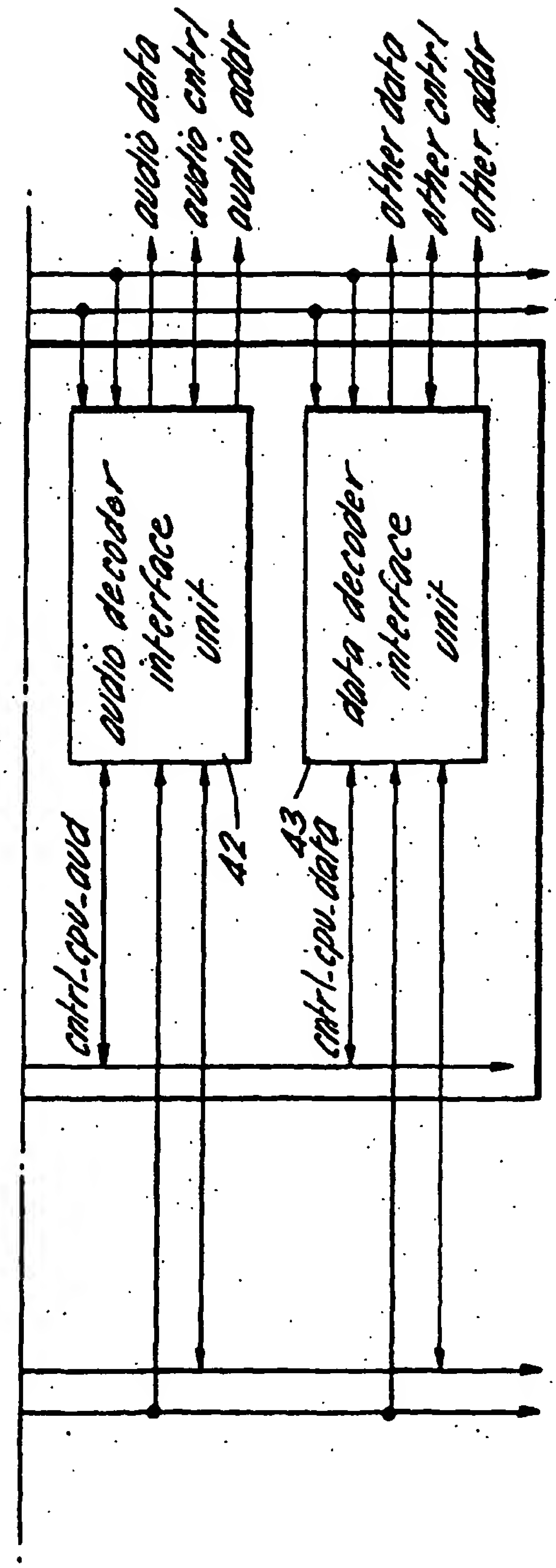

audio data
audio cntrl
audio addr
other data
other cntrl
other addr
audio decoder interface unit
data decoder interface unit
cntrl-cpu-aud
cntrl-cpu-data
42
43

FIG. 4(c)

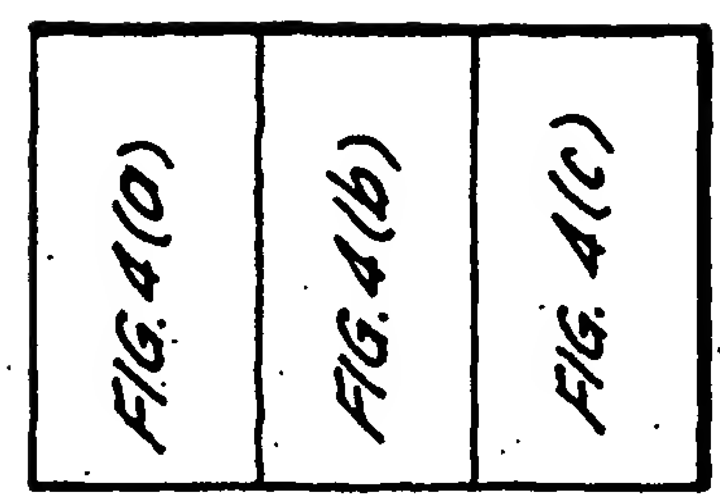


FIG. 4(a)
FIG. 4(b)
FIG. 4(c)

FIG. 4

FIG.5.

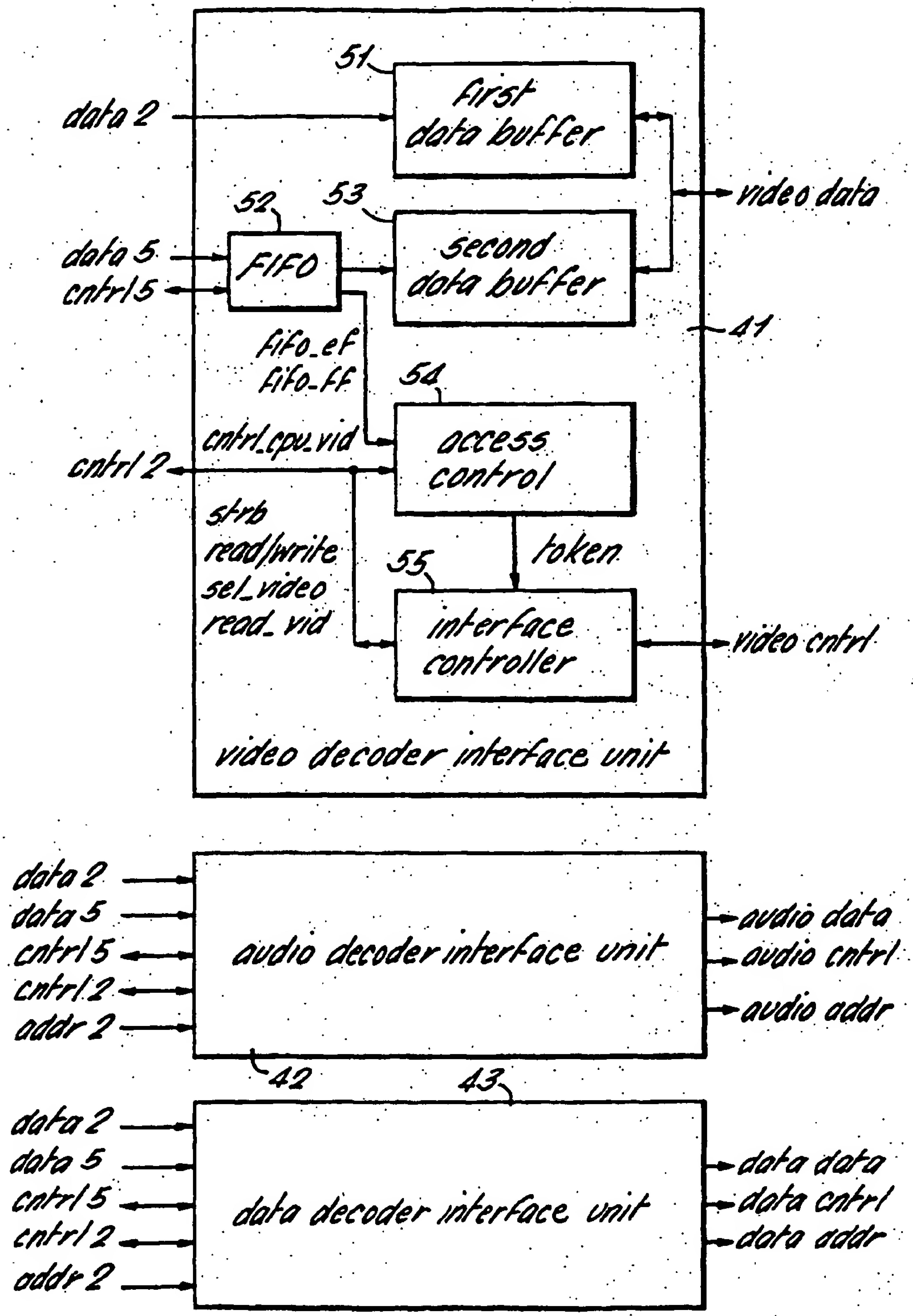
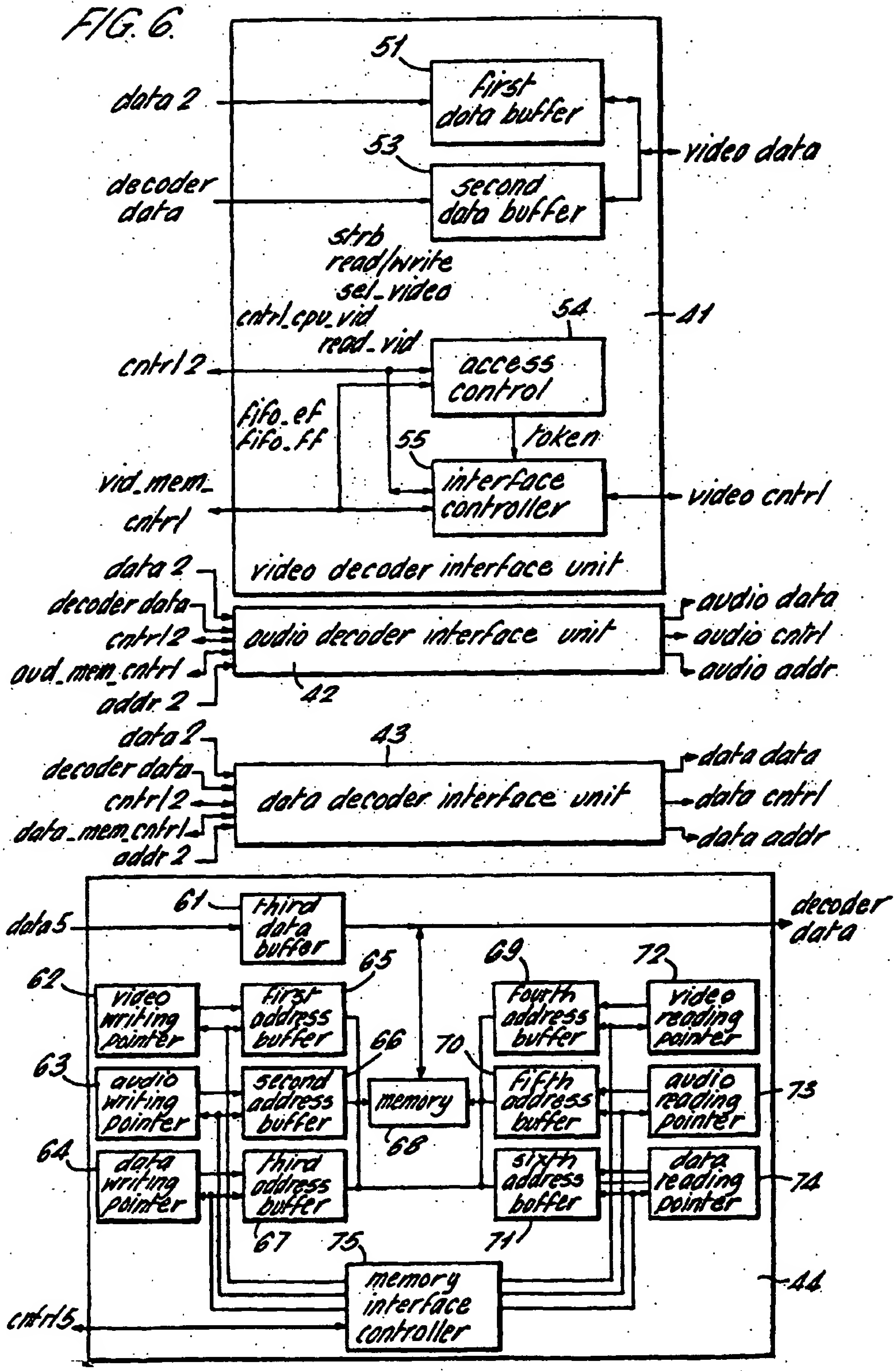

51
first
data buffer
data 2
52
53
video data
data 5
cntrl5
FIFO
second
data buffer
41
fifo_ef
fifo_ff
54
cntrl_cpu_vid
access
control
cntrl 2
strb
read/write
sel_video
read_vid
55
token
interface
controller
video cntrl
video decoder interface unit
data 2
data 5
cntrl 5
cntrl 2
addr 2
audio decoder interface unit
audio data
audio cntrl
audio addr
42
43
data 2
data 5
cntrl 5
cntrl 2
addr 2
data decoder interface unit
data data
data cntrl
data addr

FIG. 6.



51
first data buffer
53
second data buffer
data 2
decoder data
video data
strb
read/write
sel_video
cntrl_cpu_vid
read_vid
54
41
cntrl 2
access control
fifo_ef
fifo_ff
55
token
vid_mem_cntrl
interface controller
video cntrl
video decoder interface unit
data 2
decoder data
cntrl2
aud_mem_cntrl
addr 2
audio decoder interface unit
audio data
audio cntrl
audio addr
42
43
data 2
decoder data
cntrl2
data_mem_cntrl
addr 2
data decoder interface unit
data data
data cntrl
data addr
61
data5
third data buffer
decoder data
62
video writing pointer
65
first address buffer
69
fourth address buffer
72
video reading pointer
63
audio writing pointer
66
second address buffer
memory
68
70
fifth address buffer
audio reading pointer
73
64
data writing pointer
third address buffer
67
sixth address buffer
71
data reading pointer
74
75
memory interface controller
cntrl5
44

FIG. 7.

<table>
<tr><th>Syntax</th><th>No. of bits</th><th>Mnemonic</th></tr>
<tr><td>transport_packet(){</td><td></td><td></td></tr>
<tr><td>sync_byte</td><td>8</td><td>bslbf</td></tr>
<tr><td>transport_error_indicator</td><td>1</td><td>bslbf</td></tr>
<tr><td>payload_unit_start_indicator</td><td>1</td><td>bslbf</td></tr>
<tr><td>transport_priority</td><td>1</td><td>bslbf</td></tr>
<tr><td>PID</td><td>13</td><td>uimsbf</td></tr>
<tr><td>transport_scrambling_control</td><td>2</td><td>bslbf</td></tr>
<tr><td>adaptation_field_control</td><td>2</td><td>bslbf</td></tr>
<tr><td>continuity_counter</td><td>4</td><td>uimsbf</td></tr>
<tr><td>if(adaptation_field_control='10' || adaptation_field_control='11'){</td><td></td><td></td></tr>
<tr><td>adaptation_field()</td><td></td><td></td></tr>
<tr><td>}</td><td></td><td></td></tr>
<tr><td>if(adaptation_field_control='01' || adaptation_field_control='11'){</td><td></td><td></td></tr>
<tr><td>for(i=0;i<N;i++){</td><td></td><td></td></tr>
<tr><td>data_byte</td><td>8</td><td>bslbf</td></tr>
<tr><td>}</td><td></td><td></td></tr>
<tr><td>}</td><td></td><td></td></tr>
<tr><td>}</td><td></td><td></td></tr>
</table>

FIG. 8a (a)

Syntax	No. of Bits	Mnemonic
adaptation_field(){		
adaptation_field_length	8	uimsbf
if(adaptation_field_length>0){		
discontinuity_indicator	1	bslbf
random_access_indicator	1	bslbf
elementary_stream_priority_indicator	1	bslbf
PCR_flag	1	bslbf
OPCR_flag	1	bslbf
splicing_point_flag	1	bslbf
transport_private_data_flag	1	bslbf
adaptation_field_extension_flag	1	bslbf
if(PCR_flag ='1'){		
program_clock_reference_base	33	uimsbf
reserved	6	bslbf
program_clock_reference_extension	9	uimsbf
}		
if(OPCR_flag ='1'){		
original_program_clock_reference_base	33	uimsbf
reserved	6	bslbf
original_program_clock_reference_extension	9	uimsbf
}		
if(splicing_point_flag ='1'){		
splice_countdown	8	tcimsbf

```
    }
    if(transport_private_data_flag='1'){
transport      [ transport_private_data_length                          8   uimsbf
private_data   | for(i=0; i<transport_private_data_length; i--){
               |         private_data_byte                              8   bslbf
               [ }
    }
    if(adaptation_field_extension_flag='1'){
         adaptation_field_extension_length                              8   uimsbf
         ltw_flag                                                       1   bslbf
         piecewise_rate_flag                                            1   bslbf
```

FIG. 8a (b)

FIG. 8a.

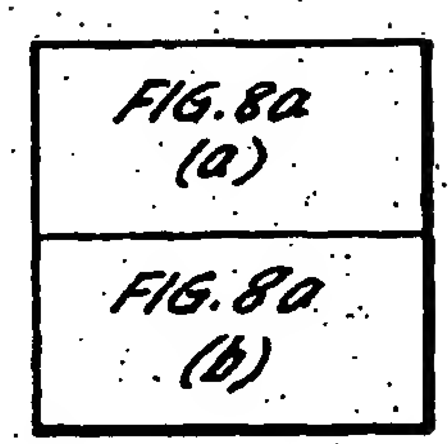


FIG. 8b (a)

Syntax		No. of Bits	Mnemonic
adapation_extension_data	seamless_splice_flag	1	bslbf
	reserved	5	bslbf
	if(ltw_flag='1'){		
	ltw_valid_flag	1	bslbf
	ltw_offset	15	uimsbf
	}		
	if(piecewise_rate_flag='1'){		
	reserved	2	bslbf
	piecewise_rate	22	uimsbf
	}		
	if(seamless_splice_flag='1'){		
	splice_type	4	bslbf
	DTS_next_au [32..30]	3	bslbf
	marker_bit	1	bslbf
	DTS_next_au [29..15]	15	bslbf
	marker_bit	1	bslbf
	DTS_next_au [14..0]	15	bslbf
	marker_bit	1	bslbf

```
            }
            for (i=0;i<N;i++){
                    reserved                                   8    bslbf
            }
      }
      for (i=0;i<N;i++){
            stuffing_byte                                      8    bslbf
      }
    }
}
```

FIG. 8b (b).

FIG. 8b.

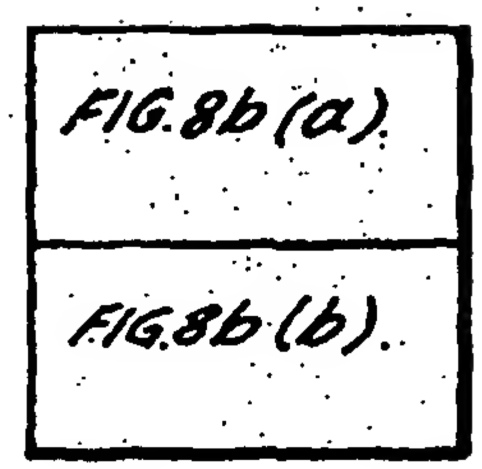


FIG.9a (a)

Syntax	No. of Bits	Mnemonic
PES_packet() {		
packet_start_code_prefix	24	bslbf
stream_id	8	uimsbf
PES_packet_length	16	uimsbf
if(stream_id != program_stream_map		
&& stream_id != padding_stream		
&& stream_id != private_stream_2)		
&& stream_id != ECM		
&& stream_id != EMM		
stream_id != program_stream_directory) {		
'10'	2	bslbf
PES_scrambling_control	2	bslbf
PES_priority	1	bslbf
data_alignment_indicator	1	bslbf
copyright	1	bslbf
original_or_copy	1	bslbf
PTS_DTS_flags	2	bslbf
ESCR_flag	1	bslbf
ES_rate_flag	1	bslbf
DSM_trick_mode_flag	1	bslbf
additional_copy_info_flag	1	bslbf
PES_CRC_flag	1	bslbf

PES_extension_flag	1	bslbf
PES_header_data_length	8	uimsbf
if(PTS_DTS_flags = '10'){		
'0010'	4	bslbf
PTS [32.30]	3	bslbf
marker_bit	1	bslbf
PTS [29.15]	15	bslbf
marker_bit	1	bslbf
PTS [14..0]	15	bslbf
marker_bit	1	bslbf
}if(PTS_DTS_flags = '11'){		
'0011'	4	bslbf

FIG. 9a (b)

FIG. 9a (a)
FIG. 9a (b)

FIG. 9a

FIG.9b (a)

Syntax	No. of Bits	Mnemonic
PTS[32..30]	3	bslbf
marker_bit	1	bslbf
PTS [29..15]	15	bslbf
marker_bit	1	bslbf
PTS [14..0]	15	bslbf
marker_bit	1	bslbf
'0001'	4	bslbf
DTS [32..30]	3	bslbf
marker_bit	1	bslbf
DTS [29..15]	15	bslbf
marker_bit	1	bslbf
DTS [14..0]	15	bslbf
marker_bit	1	bslbf
}if(ESCR_flag =='1'){		
reserved	2	bslbf
ESCR_base[32..30]	3	bslbf
marker_bit	1	bslbf
ESCR_base[29..15]	15	bslbf
marker_bit	1	bslbf
ESCR_base[14..0]	15	bslbf
marker_bit	1	bslbf

	ESCR_extension	9	uimsbf
	marker_bit	1	bslbf
	}if(ES_rate_flag='1'){		
	marker_bit	1	bslbf
	ES_rate	22	uimsbf
	marker_bit	1	bslbf
	}if(DSM_trick_mode_flag='1'){		
DSM_trick_mode_data	trick_mode_control	3	uimsbf
	if(trick_mode_control='000'){		
	field_id	2	bslbf
	intra_slice_refresh	1	bslbf
	frequency_truncation	2	bslbf
	}else if(trick_mode_control='001'){		
	field_rep_cntrl	5	uimsbf
	}else if(trick_mode_control='010'){		
	field_id	2	uimsbf
	reserved	3	bslbf
	}else if(trick_mode_control='011'){		
	field_id	2	bslbf
	intra_slice_refresh	1	bslbf
	frequency_truncation	2	bslbf
	else if(trick_mode_control='100'){		
	field_rep_cntrl	5	uimsbf

FIG.9b
(b)

FIG.9b (c)

```
        }else
            reserved                                   5    bslbf
    }if(additional_copy_info_flag = '1'){
        marker_bit                                     1    bslbf
        additional_copy_info                           7    bslbf
```

FIG.9b.

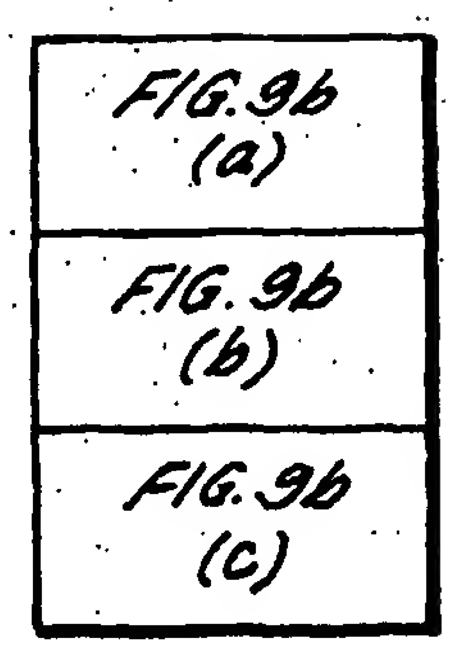


FIG. 9C (a)

Syntax	No. of Bits	Mnemonic
} if(PES_CRC_flag=='1'){		
previous_PES_packet_CRC	16	bslbf
} if(PES_extension_flag=='1'){		
PES_private_data_flag	1	bslbf
pack_header_field_flag	1	bslbf
program_packet_sequence_counter_flag	1	bslbf
P_STD_buffer_flag	1	bslbf
reserved	3	bslbf
PES_extension_flag 2	1	bslbf
if(PES_private_data_flag=='1'){		
PES_private_data	128	bslbf
}		
if(pack_header_field_flag=='1'){		
pack_field_length	8	uimsbf
pack_header()		
} if(program_packet_sequence_counter_flag=='1'){		
marker_bit	1	bslbf
program_packet_sequence_counter	7	uimsbf
marker_bit	1	bslbf
MPEG1_MPEG2 identifier	1	bslbf
original_stuff_length	6	uimsbf

PES extension_data

FIG.9c (b)

```
}if(P-STD_buffer_flag='1') {
        '01'                                                2    bslbf
        P-STD_buffer_scale                                  1    bslbf
        P-STD_buffer_size                                   13   uimsbf
}if(PES_extension_flag_2='1') {
        marker_bit                                          1    bslbf
        PES_extension_field_length                          7    uimsbf
        for(i=0;i<PES_extension_field_length;i++){
                reserved                                    8    bslbf
        }
}
}for(i=0;i<N1;i++){
        stuffing_byte                                       8    bslbf
}for(i=0;i<N2;i++){
        PES_packet_data_byte                                8    bslbf
}
}else if(stream_id=program_stream_map
||stream_id=private_stream_2
||stream_id=ECM
||stream_id=EMM
stream_id=program_stream_directory){
        for(i=1);i<PES_packet_length;i++){
                PES_packet_data_byte                        8    bslbf
```

FIG.9C (c)

```
        }
    } else if (stream_id == padding_stream) {
        for (i=0; i<N3; i++) {
            padding byte                                    8    bslbf
```

FIG.9c. (a)
FIG.9c. (b)
FIG.9c. (c)

FIG.9c.

FIG. 10a.

Syntax	No. of bits	Mnemonic
program_association_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for (i=0; i<N; i++){		
program_number	16	uimsbf
reserved	3	bslbf
if(program_number='0'){		
network_PID	13	uimsbf
}		
else {		
program_map_PID	13	uimsbf
}		
}		
CRC_32	32	rpchof
}		

FIG. 10b
(a)

Syntax	No. of bits	Mnemonic
TS_program_map_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
program_number	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved	3	bslbf
PCR_PID	13	uimsbf
reserved	4	bslbf
program_info_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}for(i=0;i<P;i++){		
stream_type	8	uimsbf

```
        reserved                                   3   bslbf
        elementary_PID                            13   uimsnf
        reserved                                   4   bslbf
        ES_info_length                            12   uimsbf
        for(i=0;i<Q;i++){
                descriptor()
        }
    {
    CRC_32                                        32   rpchof
}
```

FIG. 10b (b)

FIG. 10b

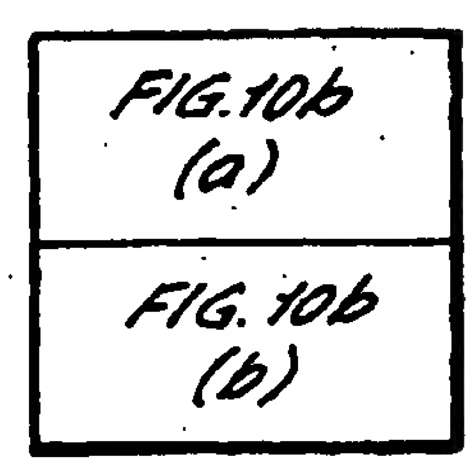


FIG. 10c.

Syntax	No. of bits	Mnemonic
CA_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
reserved	18	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0; i<N; i++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

FIG. 11.

register name	No. of bits
transport_error_indicator	1
payload_unit_start_indicator	1
transport_priority	1
PID	13
PID_V	13
PID_A	13
PID_D	13
PID_PAT	13
PID_PMT	13
PID_CAT	13
PID_NIT	13
PID_V_Flag	1
PID_A_Flag	1
PID_D_Flag	1
PID_PAT_Flag	1
PID_PMT_Flag	1
PID_CAT_Flag	1
PID_NIT_Flag	1
transport_scrambling_control	2
adaption_field_control	2
continuity_counter	4
continuity_counter_error_flag	1
tp_int_reg	11
tp_int_en	11
data_register	8

FIG.12

register name	No. of bits
PES_scrambling_control	2
PES_priority	2
data_alignment_indicator	1
copyright	1
original_or_copy	1
PTS_DTS_flags	2
ESCR_flag	1
DSM_trick_mode_flag	1
aditional_copy_info_flag	1
PES_extension_flag	1
PTS	33
DTS	33
ESCR_base	33
ESCR_extension	9
ES_rate	22
pes_int_reg	12
pes_int_en	12
data_register	8

FIG.13

register name	No. of bits
discontinuity_indicator	1
random_access_indicator	1
elementary_stream_priority_indicator	1
PCR_flag	
OPCR_flag	1
splicing_point_flag	1
transport_private_data_flag	1
adaption_field_extension_flag	1
program_clock_reference_base	33
program_clock_reference_extension	9
original_program_clock_reference_base	33
original_program_clock_reference_extension	9
spice_countdown	8
adf_int_reg	8
adf_int_en	8
data_register	8

FIG.14

```
tp_int_reg(0)⇐transport_error_indicator & tp_int_en(0);
tp_int_reg(1)⇐payload-unit-start-indicator & tp_int_en(1);
tp_int_reg(2)⇐transport_priority & tp_int_en(2);
tp_int_reg(3)⇐PID_V_flag & tp_int_en(3);
tp_int_reg(4)⇐PID_A_flag & tp_int_en(4);
tp_int_reg(5)⇐PID_D_flag & tp_int_en(5);
tp_int_reg(6)⇐PID_PAT-flag & tp_int_en(6);
tp_int_reg(7)⇐PID_PMT-flag & tp_int_en(7);
tp_int_reg(8)⇐PID_CAT_flag & tp_int_en(8);
tp_int_reg(9)⇐PID_NIT_flag & tp_int_en(9);
tp_int_reg(10)⇐(transport_scambling_control(0) &
           transport_scambling_control(1)) & tp_int_en(10);

tp int⇐tp_int_reg(0)#tp_int_reg(1)#tp_int_reg(2)#
     tp_int_reg(3)#tp_int_reg(4)#tp_int_reg(5)#
     tp_int_reg(6)#tp_int_reg(7)#tp_int_reg(8)#
     tp_int_reg(9)#tp_int_reg(10);
```

FIG.15

```
adf_int_reg(0)<=discontinuity_indicator & adf_int_en(0);
adf_int_reg(1)<=random_access_indicator & adf_int_en(1);
adf_int_reg(2)<=elementary_stream_priority_indicator & adf_int_en(2);
adf_int_reg(3)<=PCR_flag & adf_int_en(3);
adf_int_reg(4)<=OPCR_flag & adf_int_en(4);
adf_int_reg(5)<=splice_point_flag & adf_int_en(5);
adf_int_reg(6)<=transport_private_data_flag & adf_int_en(6);
adf_int_reg(7)<=adaptation_field_extension_flag & _adf_int_en(7);
     adf_int<=adf_int_reg(0)#adf_int_reg(1)#adf_int_reg(2)#
              adf_int_reg(3)#adf_int_reg(4)#adf_int_reg(5)#
              adf_int_reg(6)#adf_int_reg(7);
```

F I G.16

```
pes_int_reg(0) <= (PES_scambling_control(0)#
                   PES_scambling_control(1)) & pes_int_en(0);
pes_int_reg(1) <= PES_priority & pes_int_en(1);
pes_int_reg(2) <= data_alignment_indicator & pes_int_en(2);
pes_int_reg(3) <= copyright & pes_int_en(3);
pes_int_reg(4) <= original_or_copy & pes_int_en(4);
pes_int_reg(5) <= PTS_DTS_flags(1) & !PTS_DTS_flags(0) & pes_int_en(5);
pes_int_reg(6) <= PTS_DTS_flags(1) & PTS_DTS_flags(0) & pes_int_en(6);
pes_int_reg(7) <= ESCR_flag_ & pes_int_en(7);
pes_int_reg(8) <= DSM_trick_mode_flag & pes_int_en(8);
pes_int_reg(9) <= additional_copy_infor_flag & pes_int_en(9);
pes_int_reg(10)<=PES_CRC_flag & pes_int_en(10);
pes_int_reg(11)<=PES_extension_flag & pes_int_en(11);
     pes_int <= pes_int_reg(0)#pes_int_reg(1)# pes_int_reg(2)#
                pes_int_reg(3)#pes_int_reg(4)# pes_int_reg(5)#
                pes_int_reg(6)#pes_int_reg(7)# pes_int_reg(8)#
                pes_int_reg(9)#pes_int_reg(10)# pes_int_reg(11);
```